(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
28 August 2003 (28.08.2003)

PCT

(10) International Publication Number
WO 03/071727 A2

(51) International Patent Classification[7]: H04L

(21) International Application Number: PCT/US03/04913

(22) International Filing Date: 18 February 2003 (18.02.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/357,332	15 February 2002 (15.02.2002)	US
60/359,152	20 February 2002 (20.02.2002)	US
10/367,282	14 February 2003 (14.02.2003)	US

(71) Applicant: **MANYSTREAMS, INC.** [US/US]; 2700 Augustine Drive, Suite 178, Santa Clara, CA 95054 (US).

(72) Inventors: **PAL, Suparna**; 444 Saratoga Avenue, #7J, Santa Clara, CA 95050 (US). **DEUTSCH, Keith**; 3192 Maddux Drive, Palo Alto, CA 94303 (US).

(74) Agents: **SCHAAL, William, W.** et al.; Blakely, Sokoloff, Taylor & Zafman, 12400 Wilshire Boulevard, 7th Floor, Los Angeles, CA 90025-1026 (US).

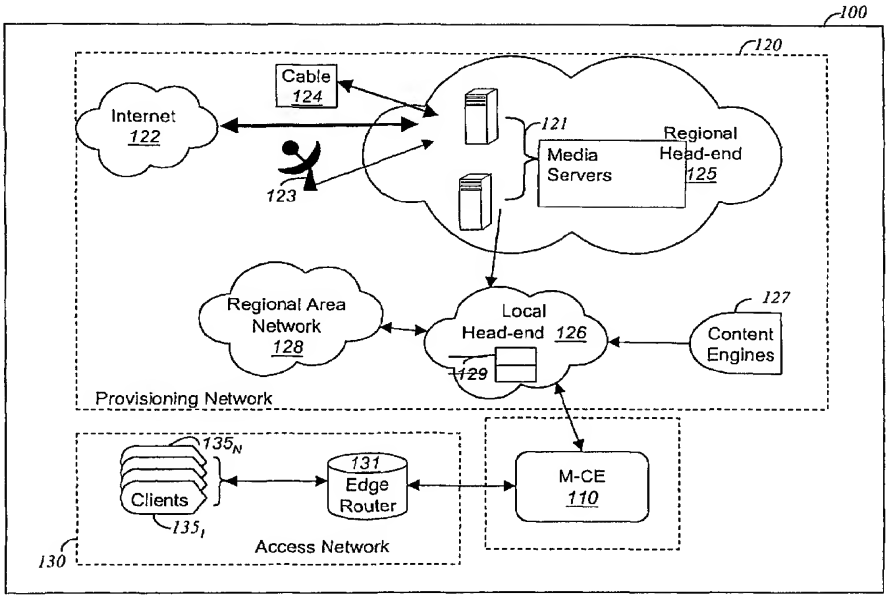
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: AN APPARATUS AND METHOD FOR THE DELIVERY OF MULTIPLE SOURCES OF MEDIA CONTENT



(57) Abstract: In one embodiment, an apparatus referred to as an intelligent media content exchange (M-CE), comprises a plurality of line cards coupled to a bus. One of the line cards is adapted to handling acquisition of at least two different types of media content from different sources. Another line card is adapted to process the at least two different types of media content in order to integrate the two different types of media content into a single stream of media content.

WO 03/071727 A2

AN APPARATUS AND METHOD FOR THE DELIVERY OF MULTIPLE SOURCES OF MEDIA CONTENT

[0001] This Application claims the benefit of priority on U.S. Provisional Patent Application No. 60/357,332 filed February 15, 2002 and U.S. Provisional Patent Application No. 60/359,152 filed February 20, 2002.

FIELD

[0002] Embodiments of the invention relate to the field of communications, in particular, to a system, apparatus and method for receiving different types of media content and transcoding the media content for transmission as a single media stream over a delivery channel of choice.

GENERAL BACKGROUND

[0003] Recently, interactive multimedia systems have been growing in popularity and are fast becoming the next generation of electronic information systems. In general terms, an interactive multimedia system provides its user an ability to control, combine, and manipulate different types of media data such as text, sound or video. This shifts the user's role from an observer to a participant.

[0004] Interactive multimedia systems, in general, are a collection of hardware and software platforms that are dynamically configured to deliver media content to one or more targeted end-users. These platforms may be designed using various types of communications equipment such as computers, memory storage devices, telephone signaling equipment (wired and/or wireless), televisions or display monitors. The most common applications of interactive multimedia systems include training programs, video games, electronic encyclopedias, and travel guides.

[0005] For instance, one type of interactive multimedia system is cable television services with computer interfaces that enable viewers to interact with television programs. Such television

programs are broadcast by high-speed interactive audiovisual communications systems that rely on digital data from fiber optic lines or digitized wireless transmissions.

[0006] Recent advances in digital signal processing techniques and, in particular, advancements in digital compression techniques, have led to new applications for providing additional digital services to a subscriber over existing telephone and coaxial cable networks. For example, it has been proposed to provide hundreds of cable television channels to subscribers by compressing digital video, transmitting the compressed digital video over conventional coaxial cable television cables, and then decompressing the video at the subscriber's set top box.

[0007] Another proposed application of this technology is a video on demand (VoD) system. For a VoD system, a subscriber communicates directly with a video service provider via telephone lines to request a particular video program from a video library. The requested video program is then routed to the subscriber's personal computer or television over telephone lines or coaxial television cables for immediate viewing. Usually, these systems use a conventional cable television network architecture or Internet Protocol (IP) network architecture.

[0008] As broadband connections acquire a larger share of online users, there will be an ever-growing need for real-time access, control, and delivery of live video, audio and other media content to the end-users. However, media content may be delivered from a plurality of sources using different transmission protocols or compression schemes such as Motion Pictures Experts Group (MPEG), Internet Protocol (IP), or Asynchronous Transfer Mode (ATM) protocol for example.

[0009] Therefore, it would be advantageous to provide a system, an apparatus and method that would be able to handle and transform various streams directed at an end-user into a single media stream.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] The invention may best be understood by referring to the following description and accompanying drawings that are used to illustrate embodiments of the invention.

[0011] Figure 1 is a schematic block diagram of the deployment view of a media delivery system in accordance with one embodiment of the invention.

[0012] Figure 2 is an exemplary diagram of screen display at a client based on media content received in accordance with one embodiment of the invention.

[0013] Figure 3 is an exemplary diagram of an intelligent media content exchange (M-CE) in accordance with one embodiment of the invention.

[0014] Figure 4 is an exemplary diagram of the functionality of the application plane deployed within the M-CE of Figure 3.

[0015] Figure 5 is an exemplary diagram of the functionality of the media plane deployed within the M-CE of Figure 3.

[0016] Figure 6 is an exemplary block diagram of a blade based media delivery architecture in accordance with one embodiment of the invention.

[0017] Figure 7 is an exemplary diagram of the delivery of plurality of media content into a single media stream targeted at a specific audience in accordance with one embodiment of the invention.

[0018] Figure 8 is an exemplary embodiment of a media pipeline architecture featuring a plurality of process filter graphs deployed the media plane in the M-CE of Figure 3.

[0019] Figure 9 is a second exemplary embodiment of a process filter graph configured to process video bit-streams within the Media Plane of the M-CE of Figure 3.

[0020] Figure 10A is a first exemplary embodiment of additional operations performed by the media analysis filter of Figure 8.

[0021] Figure 10B is a second exemplary embodiment of additional operations performed by the media analysis filter of Figure 8.

[0022] Figure 10C is a third exemplary embodiment of additional operations performed by the media analysis filter of Figure 8.

DETAILED DESCRIPTION

[0023] In general, embodiments of the invention relate to a system, apparatus and method for receiving different types of media content at an edge of the network, perhaps over different delivery schemes, and transcoding such content for delivery as a single media stream to clients over a link. In one embodiment of the invention, before transmission to a client, media content from servers are collectively aggregated to produce multimedia content with a unified framework. Such aggregation is accomplished by application driven media processing and delivery modules. By aggregating the media content at the edge of the network prior to transmission to one or more clients, any delays imposed by the physical characteristics of the network over which the multimedia content is transmitted, such as delay caused by jitter, is uniformly applied to all media forming the multimedia content.

[0024] Certain details are set forth below in order to provide a thorough understanding of various embodiments of the invention, albeit the invention may be practiced through many embodiments other than those illustrated. Well-known components and operations may not be set forth in detail in order to avoid unnecessarily obscuring this description.

[0025] In the following description, certain terminology is used to describe features of the invention. For example, a "client" is a device capable of displaying video such as a computer, television, set-top box, personal digital assistant (PDA), or the like. A "module" is software configured to perform one or more functions. The software may be executable code in the form of an application, an applet, a routine or even a series of instructions. Modules can be stored in any type of machine readable medium such as a programmable electronic circuit, a semiconductor memory device including volatile memory (e.g., random access memory, etc.) or non-volatile memory (e.g., any type of read-only memory "ROM", flash memory), a floppy diskette, an optical disk (e.g., compact disk or digital video disc "DVD"), a hard drive disk, tape, or the like.

[0026] A "link" is generally defined as an information-carrying medium that establishes a communication pathway. Examples of the medium include a physical medium (e.g., electrical wire, optical fiber, cable, bus trace, etc.) or a wireless medium (e.g., air in combination with wireless signaling technology). "Media content" is defined as information that at least comprises media data capable to being perceived by a user such as displayable alphanumeric text, audible sound, video, multidimensional (e.g. 2D/3D) computer graphics, animation or any combination thereof. In general, media content comprises media data and perhaps (i) presentation to identify the orientation of the media data and/or (ii) meta-data that describes the media data. One type of media content is multimedia content being a combination of media content from multiple sources.

[0027] Referring now to Figure 1, an illustrative block diagram of a media delivery system (MDS) 100 in accordance with one embodiment of the invention is shown. MDS 100 comprises an intelligent media content exchange (M-CE) 110, a provisioning network 120, and an access network 130. Provisioning network 120 is a portion of the network providing media content to M-CE 110, including inputs from media servers 121. M-CE 110 is normally an edge component of MDS 100 and interfaces between provisioning network 120 and access network 130.

[0028] As shown in Figure 1, for this embodiment, provisioning network 120 comprises one or more media servers 121, which may be located at the regional head-end 125. Media server(s) 121 are adapted to receive media content, typically video, from one or more of the following content transmission systems: Internet 122, satellite 123 and cable 124. The media content, however, may be originally supplied by a content provider such as a television broadcast station, video service provider (VSP), web site, or the like. The media content is routed from regional head-end 125 to a local head-end 126 such as a local cable provider.

[0029] In addition, media content may be provided to local head-end 126 from one or more content engines (CEs) 127. Examples of content engines 127 include a server that provides media content normally in the form of graphic images, not video as provided by media servers

121. A regional area network 128 provides another distribution path for media content obtained on a regional basis, not a global basis as provided by content transmission systems 122-124.

[0030] As an operational implementation, although not shown in Figure 1, a separate application server 129 may be adapted within local head-end 126 to dynamically configure M-CE 110 and provide application specific information such as personalized rich media applications based on an MPEG-4 scene graphs, i.e., adding content based on the video feed contained in the MPEG-4 transmission. This server (hereinafter referred to as "M-server") may alternatively be integrated within M-CE 110 or located so as to provide application specific information to local head-end 126 such as one of media servers 121 operating as application server 129.

For one embodiment of the invention, M-CE 110 is deployed at the edge of a broadband content delivery network (CDN) of which provisioning network 120 is a subset. Examples of such CDNs include DSL systems, cable systems, and satellite systems. Herein, M-CE 110 receives media content from provisioning network 120, integrates and processes the received media content at the edge of the CDN for delivery as multimedia content to one or more clients 135_1-135_N ($N \geq 1$) of access network 130. One function of the M-CE 110 is to operate as a universal media exchange device where media content from different sources (e.g., stored media, live media) of different formats and protocols (e.g., MPEG-2 over MPEG-2 TS, MPEG-4 over RTP, etc.) can acquire, process and deliver multimedia content as an aggregated media stream to different clients in different media formats and protocols. An illustrative example of the processing of the media content is provided below.

[0031] Access network 130 comprises an edge device 131 (e.g., edge router) in communication with M-CE 110. The edge device 131 receives multimedia content from M-CE 110 and performs address translations on the incoming multimedia content to selectively transfer the multimedia content as a media stream to one or more clients 135_1,..., and/or 135_N (generally

referred to as "client(s) 135_x) over a selected distribution channel. For broadcast transmissions, the multimedia content is sent as streams to all clients 135_1-135_N.

[0032] Referring to Figure 2, an exemplary diagram of a screen display at client in accordance with one embodiment of the invention. Screen display 200 is formed by a combination of different types of media objects. For instance, in this embodiment, one of the media objects is a first screen area 210 that displays at a higher resolution than a second screen area 220. The screen areas 210 and 220 may support real-time broadcast video as well as multicast or unicast video.

[0033] Screen display 200 further comprises 2D graphics elements. Examples of 2D graphics elements include, but are not limited or restricted to, a navigation bar 230 or images such as buttons 240 forming a control interface, advertising window 250, and layout 260. The navigation bar 230 operates as an interface to allow the end-user the ability to select what topics he or she wants to view. For instance, selection of the "FINANCE" button may cause all screen areas 210 and 220 to display selected finance programming or cause a selected finance program to be displayed at screen area 210 while other topics (e.g., weather, news, etc.) are displayed at screen area 220.

[0034] The sources for the different types of media content may be different media servers and the means of delivery to the local head-end 125 of Figure 1 may also vary. For example, video stream 220 displayed at second screen area 220 may be a MPEG stream, while the content of advertising window 250 may be delivered over Internet Protocol (IP).

[0035] Referring to both Figures 1 and 2, for this embodiment, M-CE 110 is adapted to receive from one or more media servers 121 a live news program broadcasted over a television channel, a video movie provided by a VPS, a commercial advertisement from a dedicated server or the like. In addition, M-CE 110 is adapted to receive another type of media content, such as navigator bar 230, buttons 240, layout 260 and other 2D graphic elements from content engines 127. M-CE 110 processes the different types of received media content and creates screen display 200 shown in Figure 2. The created screen display 200 is then delivered to

client(s) 135_x (e.g., television, a browser running on a computer or PDA) through access network 130.

[0036] The media content processing includes integration, packaging, and synchronization framework for the different media objects. It should be further noted that the specific details of screen display 200 may be customized on a per client basis, using a user profile available to M-CE 110 as shown in Figure 5. In one embodiment of this invention, the output stream of the M-CE 110 is MPEG-4 or an H.261 standard media stream.

[0037] As shown, layout 260 is utilized by M-CE 110 for positioning various media objects; namely screen areas 210 and 220 for video as well as 2D graphic elements 230, 240 and 250. As shown, layout 260 features first screen area 210 that supports higher resolution broadcast video for a chosen channel being displayed. Second screen area 220 is situated to provide an end-user additional video feeds being displayed, albeit the resolution of the video at second screen area 220 may be lower than that shown at first screen area 210.

[0038] In one embodiment of this invention, the displayed buttons 240 act as a control interface for user interactivity. In particular, selection of an "UP" arrow or "DOWN" arrow channel buttons 241 and 242 may alter the display location for a video feed. For instance, depression of either the "UP" or "DOWN" arrow channel buttons 241 or 242 may cause video displayed in second screen area 220 to now be displayed in first screen area 210.

[0039] The control interface also features buttons to permit rudimentary control of the presentation of the multimedia content. For instance, "PLAY" button 243 signals M-CE 110 to include video selectively displayed in first screen area 210 to be processed for transmission to the access network 130 of Figure 1. Selection of "PAUSE" button 244 or "STOP" button 245, however, signals M-CE 110 to exclude such video from being processed and integrated into screen display 200. Although not shown, the control interface may further include fast-forward and fast-rewind buttons for controlling the presentation of the media content.

[0040] It is noted that by placing M-CE 110 in close proximity to the end-user, the processing of the user-initiated signals (commands) is handled in such a manner that the latency between

an interactive function requested by the end-user and the time by which that function takes effect is extremely short.

[0041] Referring now to Figure 3, an illustrative diagram of M-CE 110 of Figure 1 in accordance with one embodiment of the invention is shown. M-CE 110 is a combination of hardware and software that is segmented into different layers (referred to as "planes") for handling certain functions. These planes include, but are not limited or restricted to two or more of the following: application plane 310, media plane 320, management plane 330, and network plane 340.

[0042] Application plane 310 provides a connection with M-server 129 of Figure 1 as well as content packagers, and other M-CEs. This connection may be accomplished through a link 360 using a hypertext transfer protocol (HTTP) for example. M-server 129 may comprise one or more XMT based presentation servers that create personalized rich media applications based on an MPEG-4 scene graph and system frameworks (XMT-O and XMT-A). In particular, application plane 310 receives and parses MPEG-4 scene information in accordance with an XMT-O and XMT-A format and associates this information with a client session. "XMT-O" and "XMT-A" is part of the Extensible MPEG-4 Textual (XMT) format that is based on a two-tier framework: XMT-O provides a high level of abstraction of an MPEG-4 scene while XMT-A provides the lower-level representation of the scene. In addition, application plane 310 extracts network provisioning information, such as service creation and activation, type of feeds requested, and so forth, and sends this information to media plane 320.

[0043] Application plane 310 initiates a client session that includes an application session and a user session for each user to whom a media application is served. The "application session" maintains the application related states, such as the application template which provides the basic handling information for a specific application, such as the fields in a certain display format. The user session created in M-CE 110 has a one-to-one relationship with the application session. The purpose of the "user session" is to aggregate different network sessions (e.g., control sessions and data sessions) in one user context. The user session and

application session communicate with each other using extensible markup language (XML) messages over HTTP.

[0044] Referring now to Figure 4, an exemplary diagram of the functionality of the application plane 310 deployed within the M-CE 110 of Figure 3 is shown. The functionality of M-CE 110 differs from traditional streaming device and application servers combinations, which are not integrated through any protocol. In particular, traditionally, an application server sends the presentation to the client device, which connects to the media servers directly to obtain the streams. In a multimedia application, strict synchronization requirements are imposed between the presentation and media streams. For example, in a distance learning application, a slide show, textual content and audio video speech can be synchronized in one presentation. The textual content may be part of application presentation, but the slide show images, audio and video content are part of media streams served by a media server. These strict synchronization requirements usually cannot be obtained by systems having disconnected application and media servers.

[0045] Herein, M-Server 129 of Figure 1 (the application server) and the M-CE 110 (the streaming gateway) are interconnected via a protocol so that the application presentation and media streams can be delivered to the client in a synchronized way. The protocol between M-Server 129 and M-CE 100 is a unified messaging language based on standard based descriptors from MPEG-4, MPEG-7 and MPEG-21 standards. The MPEG-4 provides the presentation and media description, MPEG-7 provides stream processing description such as transcoding and MPEG-21 provides the digital rights management information regarding the media content. The protocol between M-Server 129 and M-CE 110 is composed of MOML messages. MOML stands for MultiMedia Object Manipulation Language. Also, multimedia application presentation behavior changes as user interacts with the application, such as based on user interaction the video window size can increase or decrease. This drives media processing requirements in M-CE 110. For example, when the video window size decreases, the

associated video can be scaled down to save bandwidth. This causes a message, such as media processing instruction, to be sent via protocol from M-Server 129 to M-CE 110.

[0046] Application plane 310 of M-CE 110 parses the message and configures the media pipeline to process the media streams accordingly. As shown in detail in Figure 4, application plane 310 comprises an HTTP server 311, a MOML parser 312, an MPEG-4 XMT parser 313, an MPEG-7 parser 314, an MPEG-21 parser 315 and a media plane interface 316. In particular, M-server 129 transfers a MOML message (not shown) to HTTP server 311. As an illustrative embodiment, the MOML message contains a presentation section, a media processing section and a service rights management section (e.g., MPEG-4 XMT, MPEG-7 and MPEG-21 constructs embedded in the message). Of course, other configurations of the message may be used.

[0047] HTTP server 311 routes the MOML message to MOML parser 312, which extracts information associated with the presentation (e.g. MPEG-4 scene information and object descriptor "OD") and routes such information to MPEG-4 XMT parser 313. MPEG-4 XMT parser 313 generates commands utilized by media plane interface 316 to configure media plane 320.

[0048] Similarly, MOML parser 312 extracts information associated with media processing from the MOML message and provides such information to MPEG-7 parser 314. Examples of this extracted information include a media processing hint related to transcoding, transrating thresholds, or the like. This information is provided to MPEG-7 parser 314, which generates commands utilized by media plane interface 316 to configure media plane 320.

[0049] MOML parser 312 further extracts information associated with service rights management data such policies for the media streams being provided (e.g., playback time limits, playback number limits, etc.). This information is provided to MPEG-21 parser 315, which also generates commands utilized by media plane interface 316 to configure media plane 320.

[0050] Referring to Figures 3 and 5, media plane 320 is responsible for media stream acquisition, processing, and delivery. Media plane 320 comprises a plurality of modules; namely, a media acquisition module (MAM) 321, a media processing module (MPM) 322, and a media delivery module (MDM) 323. MAM 321 establishes connections and acquires media streams from media server(s) 121 and/or 127 of Figure 1 as perhaps other M-CEs. The acquired media streams are delivered to MPM 322 and/or and MDM 323 for further processing. MPM 322 processes media content received from MAM 321 and delivers the processed media content to MDM 323. Possible MPM processing operations include, but are not limited or restricted to transcoding, transrating (adjusting for differences in frame rate), encryption, and decryption.

[0051] MDM 323 is responsible for receiving media content from MPM 322 and delivering the media (multimedia) content to client(s) 135_x of Figure 1 or to another M-CE. MDM 323 configures the data channel for each client 135_1-135_N, thereby establishing a session with either a specific client or a multicast data port. Media plane 320, using MDM 323, communicates with media server(s) 121 and/or 127 and client(s) 135_x through communication links 350 and 370 where information is transmitted using Rapid Transport Protocol (RTP) and signaling is accomplished using Real-Time Streaming Protocol (RTSP).

[0052] As shown in Figure 5, media manager 324 is responsible to interpret all incoming information (e.g., presentation, media processing, service rights management) and configure MAM 321, MPM 322 and MDM 323 via Common Object Request Broker Architecture (CORBA) API 325 for delivery of media content from any server(s) 121 and/or 127 to a targeted client 135_x.

[0053] In one embodiment, MAM 321, MPM 322, and MDM 323 are self-contained modules, which can be distributed over different physical line cards in a multi-chassis box. The modules 321-323 communicate with each other using industry standard CORBA messages over CORBA API 326 for exchanging control information. The modules 321-323 use inter-process

communication (IPC) mechanisms such as sockets to exchange media content. A detailed description for such architecture is shown in Figure 6.

[0054] Management plane 330 is responsible for administration, management, and configuration of M-CE 110 of Figure 1. Management plane 330 supports a variety of external communication protocols including Signaling Network Management Protocol (SNMP), Telnet, Simple Object Access Protocol (SOAP), and Hypertext Markup Language (HTML).

[0055] Network plane 340 is responsible for interfacing with other standard network elements such as routers and content routers. Mainly, network plane 340 is involved in configuring the network environment for quality of service (QoS) provisioning, and for maintaining routing tables.

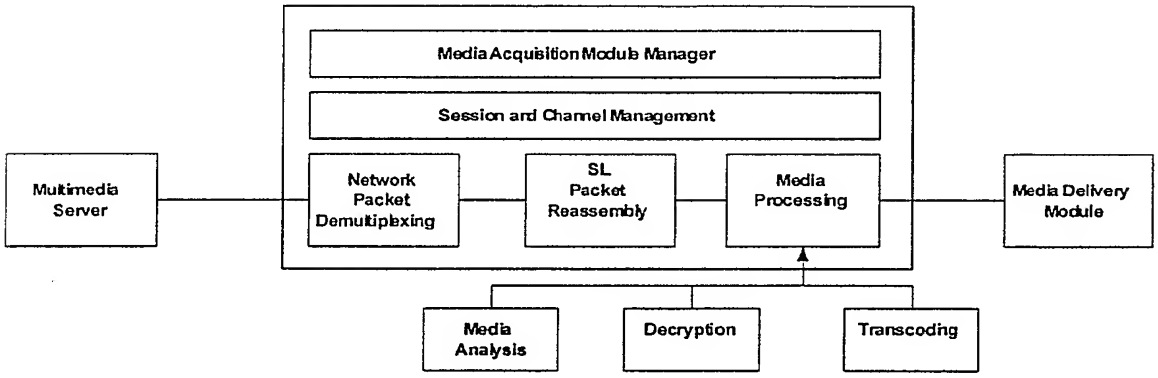
[0056] The architecture of M-CE 110 provides the flexibility to aggregate unicast streams, multicast streams, and/or broadcast streams into one media application delivered to a particular user. For example, M-CE 110 may receive multicast streams from one or more IP networks, broadcast streams from one or more satellite networks, and unicast streams from one or more video server, through different MAMs. The different types of streams are served via MDM 323 to one client in a single application context.

[0057] It should be noted that the four functional planes of M-CE 110 interoperate to provide a complete, deployable solution. However, although not shown, it is contemplated that M-CE 110 may be configured without the network 340 where no direct network connectivity is needed or without management plane 330 if the management functionality is allocated into other modules.

[0058] Referring now to Figure 6, an illustrative diagram of M-CE 110 of Figure 1 configured as a blade-based MPEG-4 media delivery architecture 400 is shown. For this embodiment, media plane 320 of Figure 3 resides in multiple blades (hereinafter referred to as "line cards"). Each line card may implement one or more modules.

- A **Unicast MAM** establishes a unicast session with a media server or a peer IMCE. The data is received in RTP format and control signaling is done using RTSP. The session is explicitly created in response to a user request for service.
- A **Multicast MAM** establishes connection and session with a multicast media data source. The data is received using RTP or MPEG-4 FlexMux. Control signaling is done according to specific multicast protocol. The session is initiated through explicit user administration, where administrator will send the multicast information to the module to initiate a session.
- A **Broadcast MAM:** The MAM establishes session and receives data from a broadcast network like a satellite network. The media data is received in either MPEG2-TS format or RTP format. The session is established via explicit user administration.

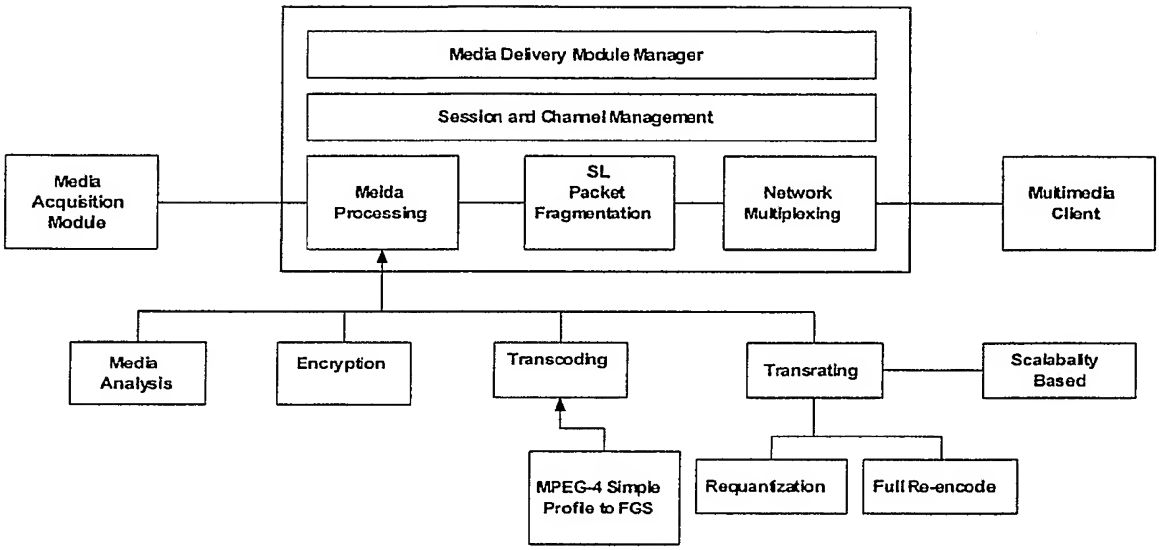
One of the key values provided by the Media Acquisition Module is to remove network jitter before delivering media data to the Media Delivery Module.



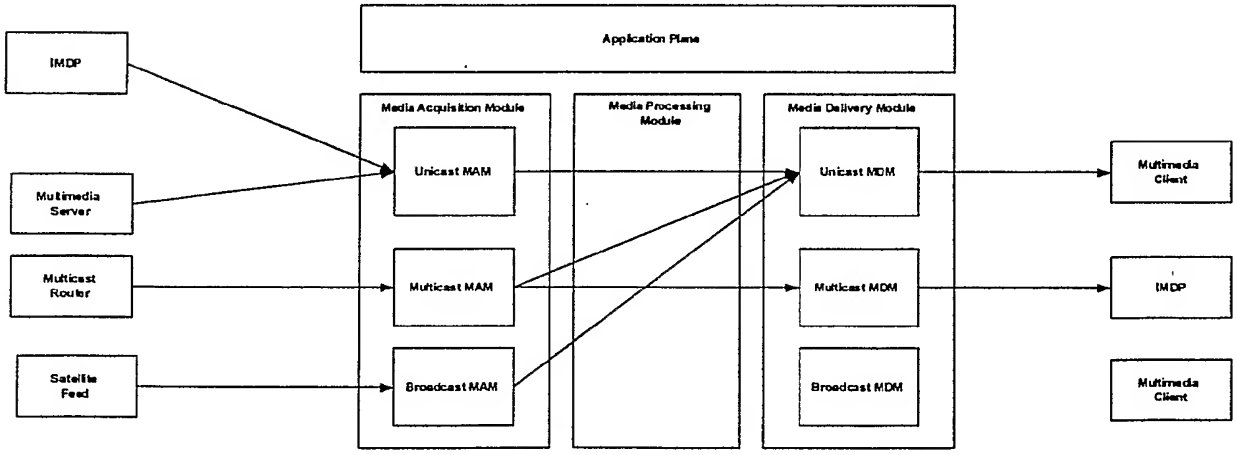
The following figure describes the high level architecture of Media Delivery Module (MDM). The MDM is responsible for receiving media data from a MAM and delivering the media data to multimedia clients or a peer IMCE. The MDM manager provides a CORBA API for sending and receiving messages to and from the Media Delivery Module. The Media Delivery Module provides a key value, which is edge media processing. The Media Delivery Module configures the data channel with a set of media processors necessary for a particular user, receiving media on a specific type of network and for a specific type of device.

Following are the different types of Media Delivery Module:

- A **Unicast MDM** establishes a session explicitly in response to a client request for service. The data is transmitted using the RTP protocol. Signaling with the client device is done using the RTSP protocol.
- A **Multicast MDM** establishes a session with a multicast data port and delivers multiplexed media to that multicast port. The data is streamed using the RTP protocol and signaling is done using multicast protocol.

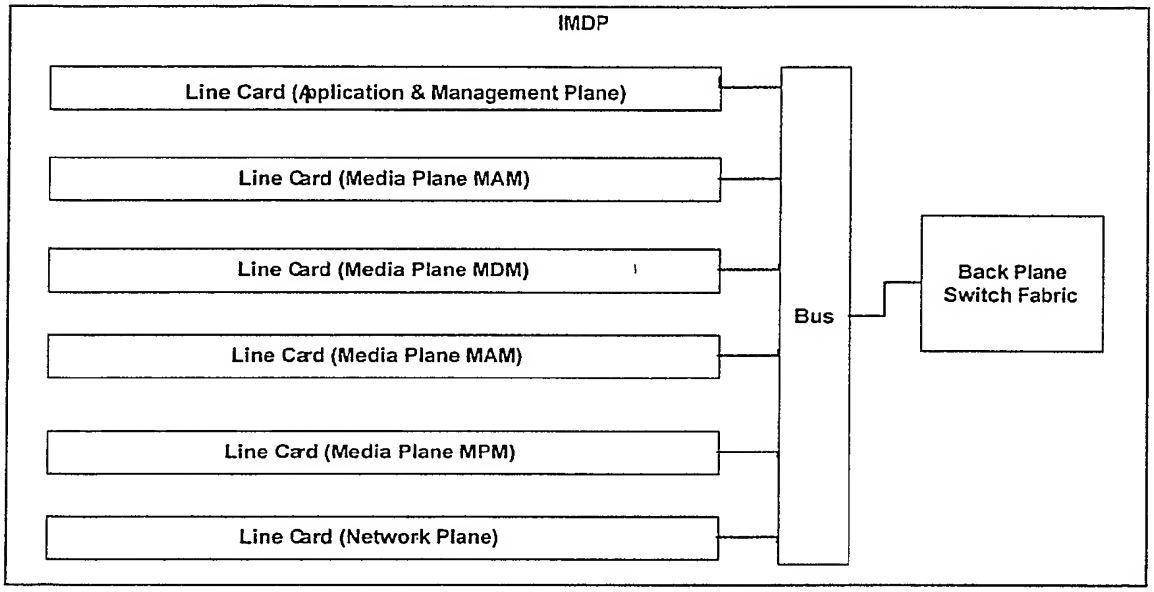


MPEG-4 MEDIA NETWORKING BRIDGE ARCHITECTURE



IMCE media plane is divided into two modules MAM and MDM. The MAM has series of network stacks, which are unicast, multicast and broadcast covering different types of transport protocol. MDM also has the similar configuration. The network sessions from MAM and MDM are aggregated under an application session in application plane. The application session in the application plane aggregates the MAM and MDM network session. So IMCE acts as a media bridge where multicast streams from IP network, broad cast streams from satellite network, and an unicast streams from a video server through different MAM and serve via unicast MDM to one client in a single application context.

BLADE-BASED MPEG-4 MEDIA DELIVERY ARCHITECTURE

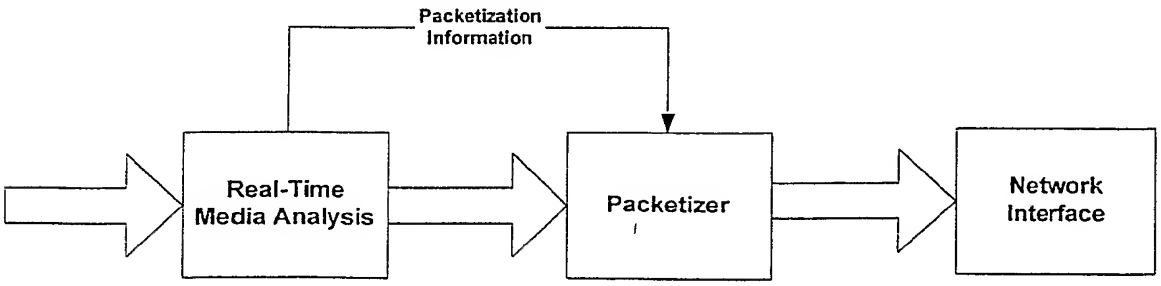


The IMCE is a blade-based MPEG-4 edge media delivery platform. The Media Plane can reside in multiple blades or line cards, where each Media Acquisition Module, Media Delivery Module and Media Processing Module resides in different line cards. Also each line card may have different capability. For example, one line card may have MPEG-2 transcoder, MPEG-2 TS media networking stack with DVB-ASI input for Media Acquisition Module and other line card may have gigabit-Ethernet input with RTP/ RTSP media network stack for the MAM. Based on the information provided during session setup, appropriate line cards will be chosen.

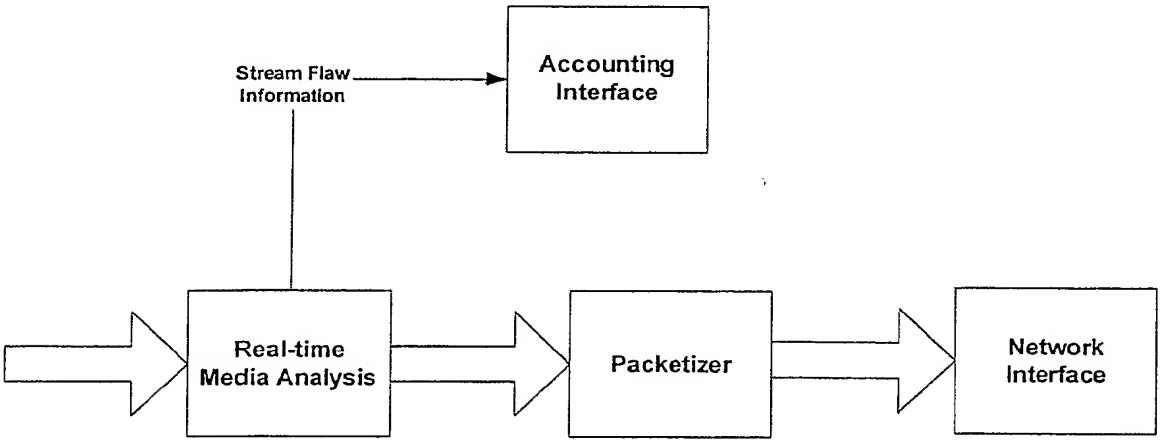
The MAM, MPM, and MDM use CORBA to exchange control information. The MAM, MPM, and MDM exchange media data using a socket, shared memory, or a pipe (any interprocess communication mechanism).

REAL-TIME MEDIA ANALYSIS IN MPEG-4 MEDIA DELIVERY PIPELINE

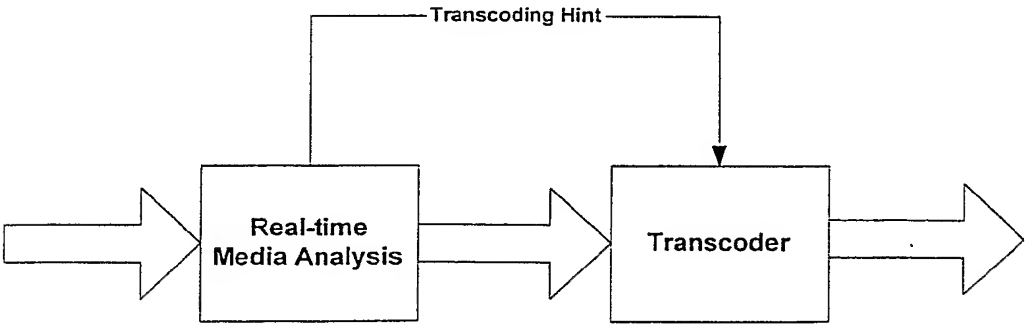
The following diagrams illustrate how a real-time media analysis processing filter can be used in the MPEG-4 media delivery pipeline. The media analysis module provides different functionalities, which are described in different figures. The concept of using real-time media analysis filter in a MPEG-4 media delivery pipeline is a unique idea.



The data pipeline will have a filter called Real-Time Media Analysis. This filter will parse the video bit-stream in real-time and will generate information such as VOL (MPEG4 Video Object Layer) boundary or macro-block boundary. This information is used by the segmentation engine of the packetizer to segment the access unit. Considering slice boundary, VOL boundary, macro-block boundary in access unit segmentation ensures that video can be reconstructed more accurately with greater quality in case of packet loss in the network. This type of Media Analysis filter will be used in an MDM. The media will be of nature MPEG-4.

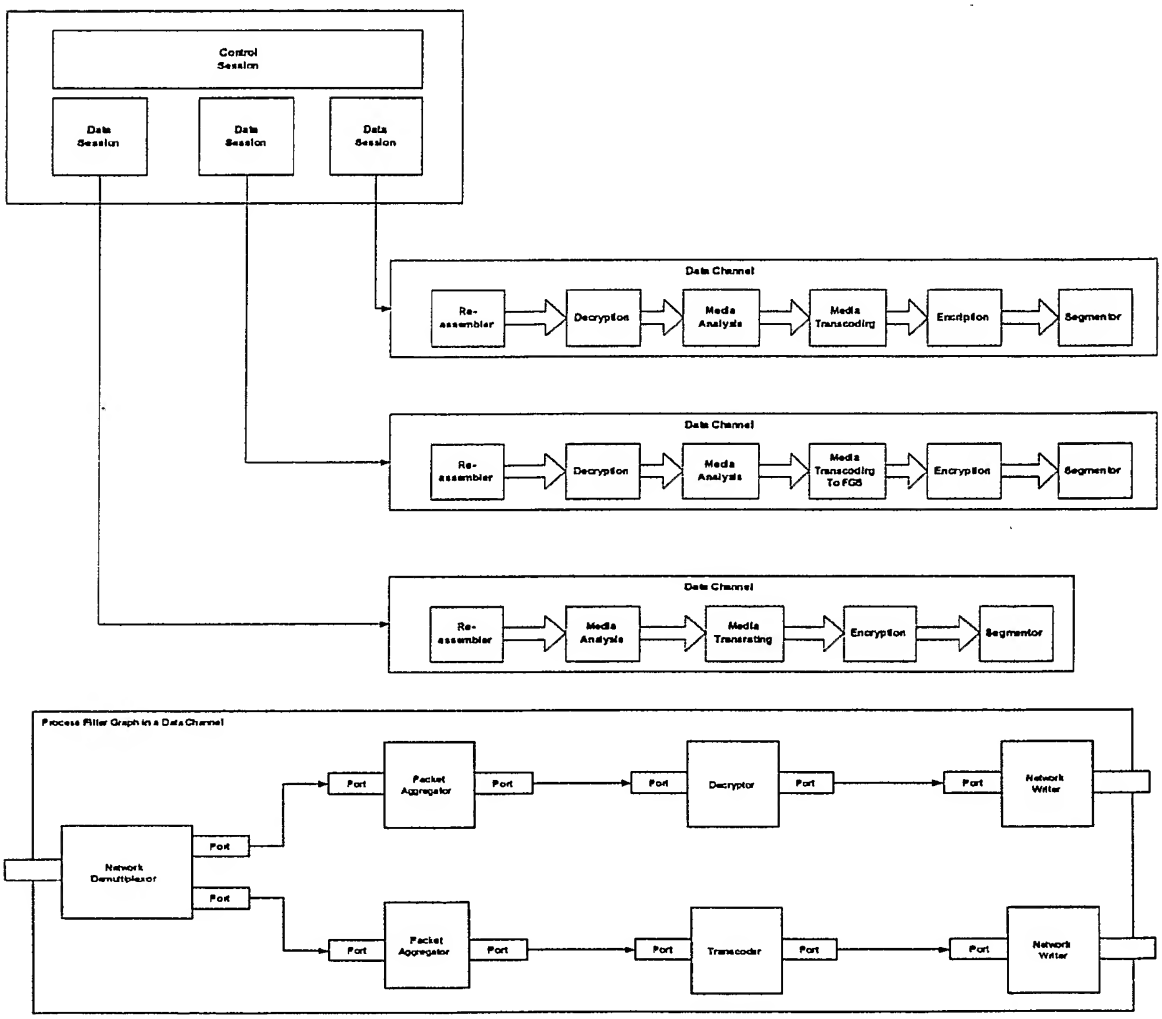


The real-time media analysis filter also provides stream flaw detection. It parses the incoming media stream and finds flaws in encoding. If it detects any flaw, it reports to the Accounting Interface. This information is logged and can be provided to the media content provider. Also this information can be transmitted to any real-time encoder for the purpose of adjusting the encoding parameters to avoid stream flaws, if the media source is a real-time encoder. This type of real-time media analysis filter is used in the MAM, where media is received either from a remote media server or from a real-time encoder. The media is encoded, formatted, and packaged as MPEG-4.



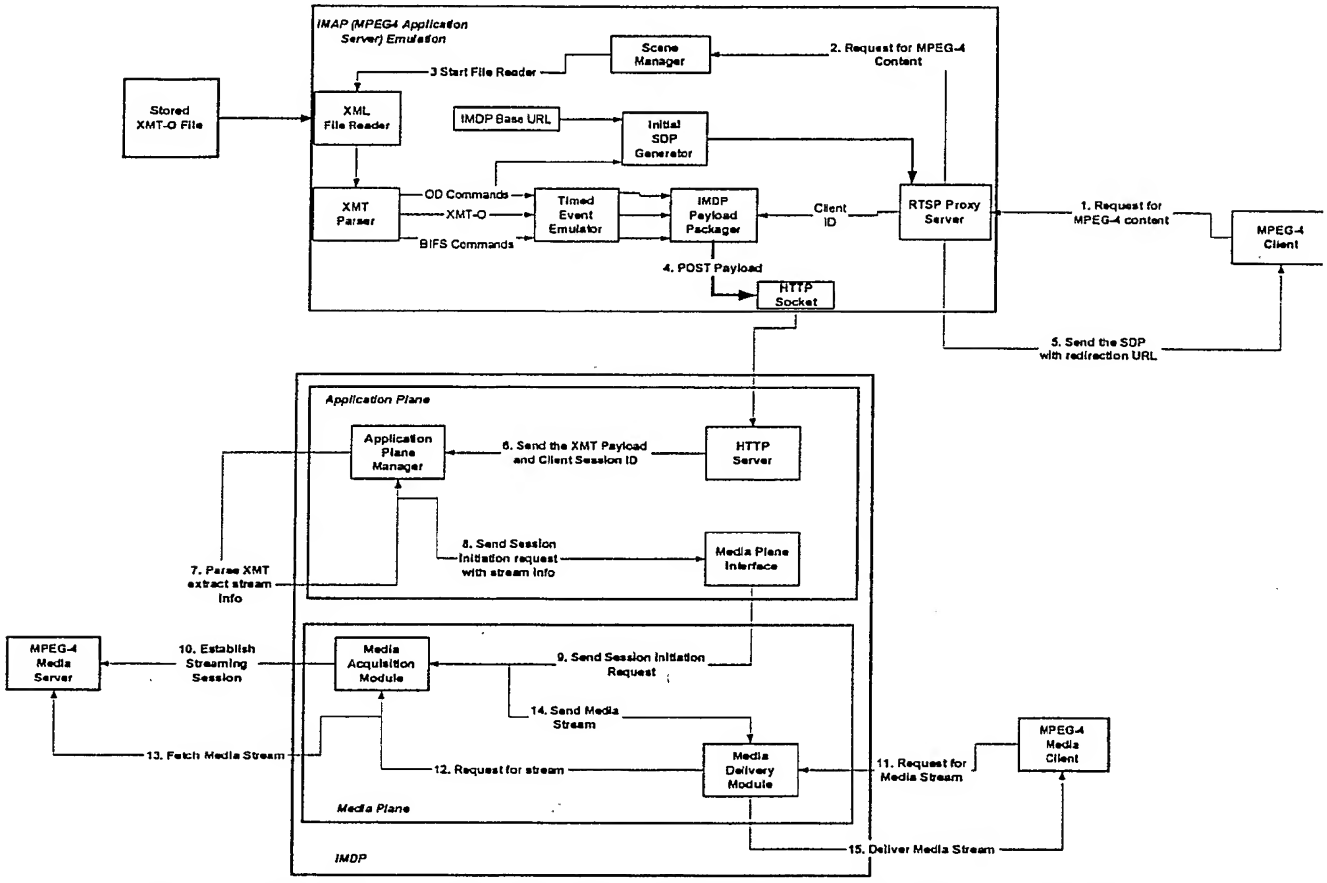
The real-time media analysis filter can be used to generate transcoding hint information. The real-time media analysis filter will parse the video bit-stream to understand the complexity of the scene, scene change and fade detection, organization of DCT coefficients etc. Based on this information it will generate transcoding hint information. The transcoding hint provides information, which is interpreted by a real-time transcoder to transrate the bit-stream for certain bandwidth condition. This type of media analysis filter will be used in the data pipeline of Media Delivery Module along with real-time transcoder and transrater. The video-bit stream will be of type MPEG-4 compliant video and format of the transcoding hint information will be of format MPEG-7.

A DYNAMICALLY CONFIGURABLE PROCESSING FILTER GRAPH-BASED MPEG-4 MEDIA DELIVERY PIPELINE



The heart of data channel is the process filter graph. The process filter graph is a sequence of processing filters connected with each other via a port. The port can be a socket or shred buffer. The processing filters are active elements in a sense they can run in their own thread context. This process filter graph topology is determined at the time of data session establishment. This makes the MPEG-4 media pipeline completely programmable. For example, in case of transmission of scalable video from a server the base layer can be encrypted, but the enhanced layers can carry clear media. So the process filter sequence for handling the base layer video stream will be different than enhanced layer video stream. The above-mentioned figure illustrates the difference. This programmable MPEG-4 media pipeline can be changed dynamically. Also due to a change in the scene, it may be necessary to insert a new filter. So this programmable media pipeline makes the IMCE application-aware dynamically configurable MPEG-4 edge network device.

IMAP EMULATION, DATA FLOW, & SESSION INITIATION



The above diagram describes overall data and control flow for delivering MPEG-4 content. The following steps describes in detail in control and data flow.

- The MPEG-4 client establishes an RTSP connection with the IMAP (BIFS Application Server) by sending a connect request. The RTSP proxy server in IMAP intercepts the request and creates an RTSP network connection with MPEG-4 client. The MPEG-4 client then requests MPEG-4 content through a RTSP describe request.
- The RTSP proxy server in IMAP receives the request and forwards it to the IMAP scene manager.
- The scene manager fetches the pre-authored XMT-O scene description through an MPEG-4 file reader.
- The XMT-O is parsed and processed to produce groups of time stamped OD and BIFS Commands and associated XMT-O.
- The initial SDP is generated from the OD Commands and IMCE URL, and returned to the MPEG-4 client via the RTSP proxy server. The SDP contains the IMCE IP address from where MPEG-4 client will directly fetch the media streams.
- The Initial BIFS & OD Command and XMT-O Content are passed to the IMCE Payload Packager where they are combined with the Client ID and Session Initiation request, and POSTed to the IMCE via an HTTP socket
- The HTTP server in application plane receives the POST, extracts the XMT representation and

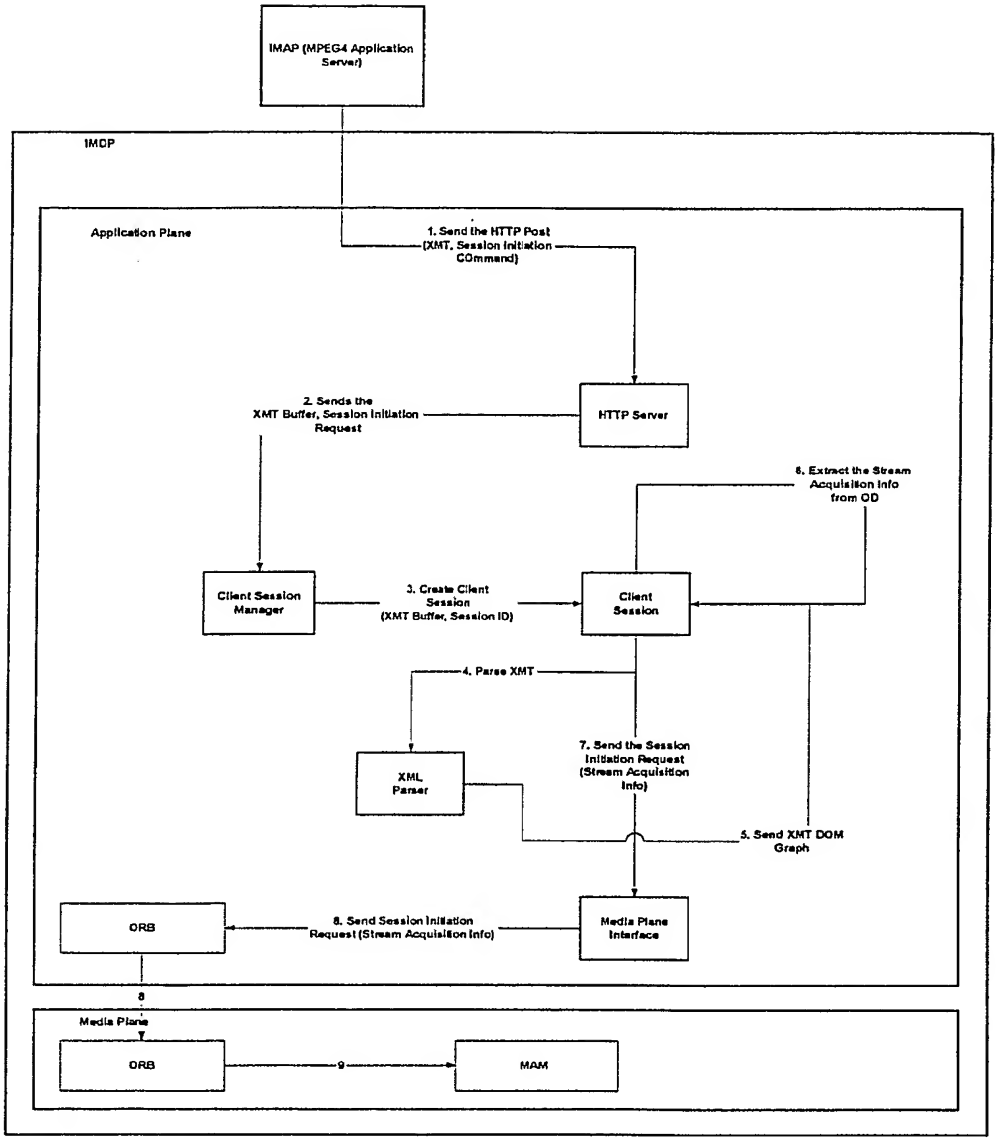
session initiation request. Then sends the session initiation request to application plane manager. The application plane manager parses the XMT representation and extracts the stream acquisition information from OD. The stream acquisition information contains the stream id and MPEG-4 streaming server location. The application plane manager sends a session initiation request to Media Plane Interface. The Media Plane Interface provides interface between the application plane and Media Plane.

- The Media Plane Interface sends a session initiation request (with stream acquisition information) to the Media Acquisition Module. The Media Acquisition Module establishes connection and session with the media servers to receive the media stream. The connection and session establishment is done through RTSP protocol.
- The MPEG-4 client initiates a connection and session with Media Delivery Module using RTSP protocol. Then it requests for media stream from Media Delivery Module. The Media Delivery Module requests for media streams from Media Acquisition Module. The Media Acquisition Module fetches the media stream over established connection and session from media servers and delivers the streams to Media Delivery Module. The Media Delivery Module delivers the media streams to MPEG-4 client.
- In the IMAP, the Timed Event Emulator fires a series of events timed to precede the BIFS & OD Command timestamps by an amount that exceeds the latency of the entire system. Each such event causes the relevant Command AU and XMT-O to be packaged with the Client ID and POSTed to the IMCE for processing as a session update.

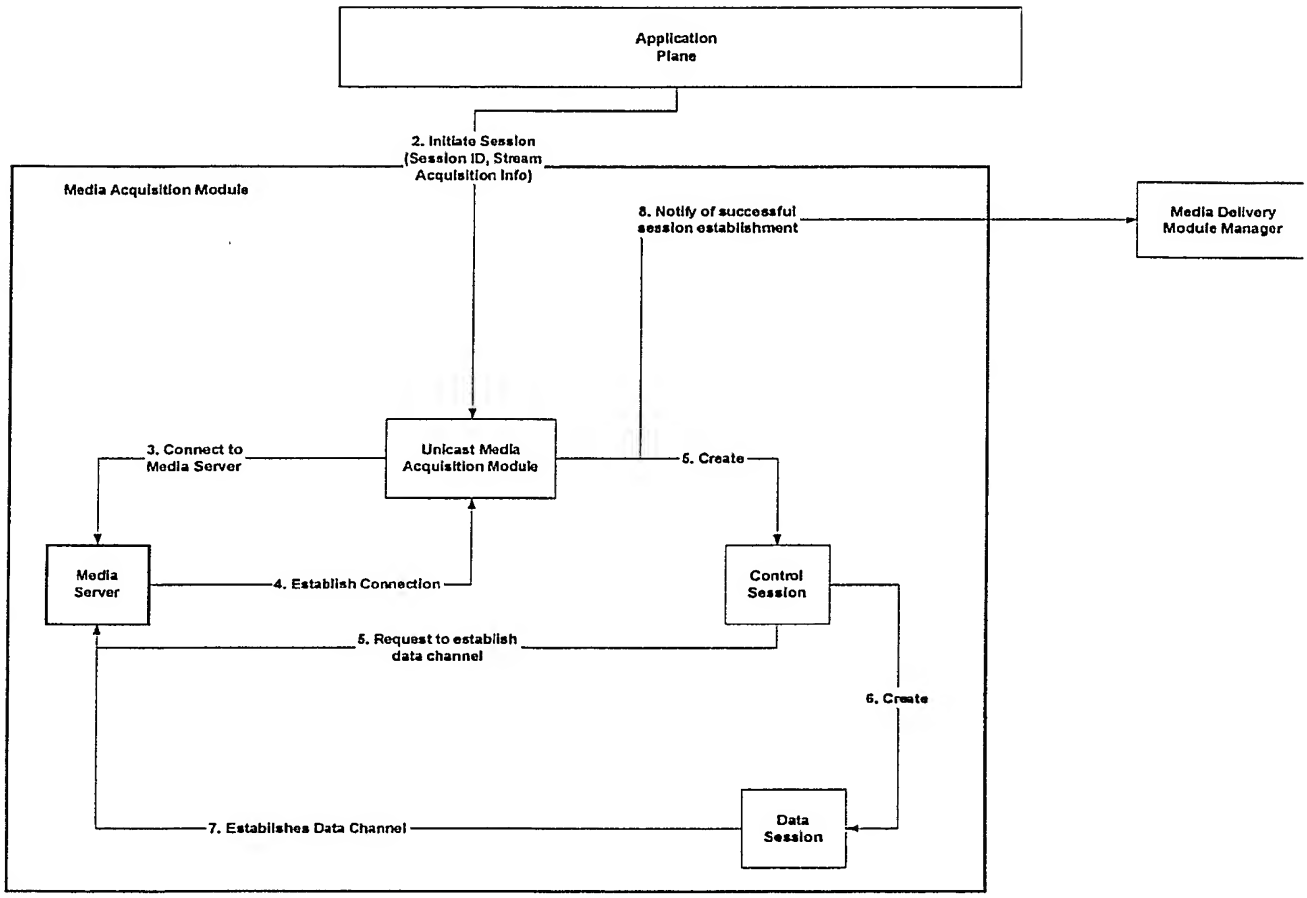
Detail Data and Control Flow in Application Plane for Session Initiation

The following diagram explains the major components in the application plane and the data and control flow of the session initiation request. The control and data flow of the session initiation handling is described in the flowing steps.

- The HTTP server of the application plane receives the POSTed XML package and the session initiation request sent by IMAP. The application plane resides in IMCE.
- The HTTP server extracts the XML Payload from the HTTP packet and sends it to the application plane manager. The application plane client session manager creates a client session and passes the XMT Buffer and the session initiation request parameters.
- The client session parses the XMT buffer, extracts the stream acquisition information from the OD. Then it sends a session initiation request to the Media Plane Interface. The session initiation request will have the Stream Acquisition Information as argument. The stream acquisition information contains the MPEG-4 media server IP address and the stream Ids. The Media Plane Interface serves as an interface between application plane and Media Plane. All the request and response between application plane and Media Plane goes through the Media Plane Interface.
- The Media Plane Interface sends the session initiation request to appropriate MAM (Media Acquisition Module) via CORBA API. In the demo scenario, both application plane and Media Plane will run in one processor, so one ORB will reside in the machine. The MAM resides logically in Media Plane.


IMAP (MPEG4 Application Server)
IMDP
Application Plane
1. Send the HTTP Post (XMT, Session Initiation COmmand)
2. Sends the XMT Buffer, Session Initiation Request
HTTP Server
6. Extract the Stream Acquisition Info from OD
Client Session Manager
3. Create Client Session (XMT Buffer, Session ID)
Client Session
4. Parse XMT
XML Parser
7. Send the Session Initiation Request (Stream Acquisition Info)
5. Send XMT DOM Graph
ORB
8. Send Session Initiation Request (Stream Acquisition Info)
Media Plane Interface
8
Media Plane
ORB
9
MAM

Session Initiation in Media Acquisition Module



The above-mentioned diagram explains the control and data flow associated with the session initiation process.

- The Media Plane Interface of the application plane sends a session initiation request to the Unicast Media Acquisition Module. This session id and stream acquisition information is passed as argument. This session initiation is a CORBA method invocation. The Media Plane Interface selects a type of Media Acquisition Module based on information send by IMAP in stream acquisition info.
- The Unicast Media Acquisition Module then sends a connect request to a well-known RTSP port of the media server and establishes a successful connection with the media server. Then Unicast Media Acquisition Module creates a control session and passes the network connection handle to the control session.
- The control session sends RTSP setup request to the server to establish data session. It creates a data session and effectively establishes data channel after getting successful response from the media server.
- The Unicast Media Acquisition Module then notifies the corresponding Media Delivery Module

of successful session establishment.

ARCHITECTURAL GOALS AND CONSTRAINTS

This section describes the key assumptions and system constraints, which has significant architectural bearing in the architecture.

The output of the Media Acquisition Module will be an Access Unit because of the following reasons

- The media can be delivered from the origin server over different protocol such as MPEG-2 TS, which may not carry SL packets. So we cannot constrain the system by the assumption that Media Acquisition Module always will send SL packets.
- The Media Processing Module can be distributed over different boards to avoid processing overload on a particular board. For example the transcoding may take place in one board but the encryption may take place in another board, where these processing will be done on AU. In that case input and output to and from the Media Processing Module s needs to be standardized. So by standardizing to be AU provides more advantage over SL packets.

The Media Acquisition Module will support FTP/ HTTP file download protocol for acquiring media from origin servers but this is not the most convenient way to get media for the following reasons

- FTP/ HTTP file download protocol cannot be used for live media.
- In order to download and deliver media in a synchronized and real-time fashion, the media needs to be stored in the file in a time based interleaved way. This requires prior knowledge of how the media is stored in the file.
- Different media tracks can be distributed over different physical files. In case of a content where multiple languages are supported, the media for different languages can be stored in different files. So file download can provide significant problem for media acquisition.

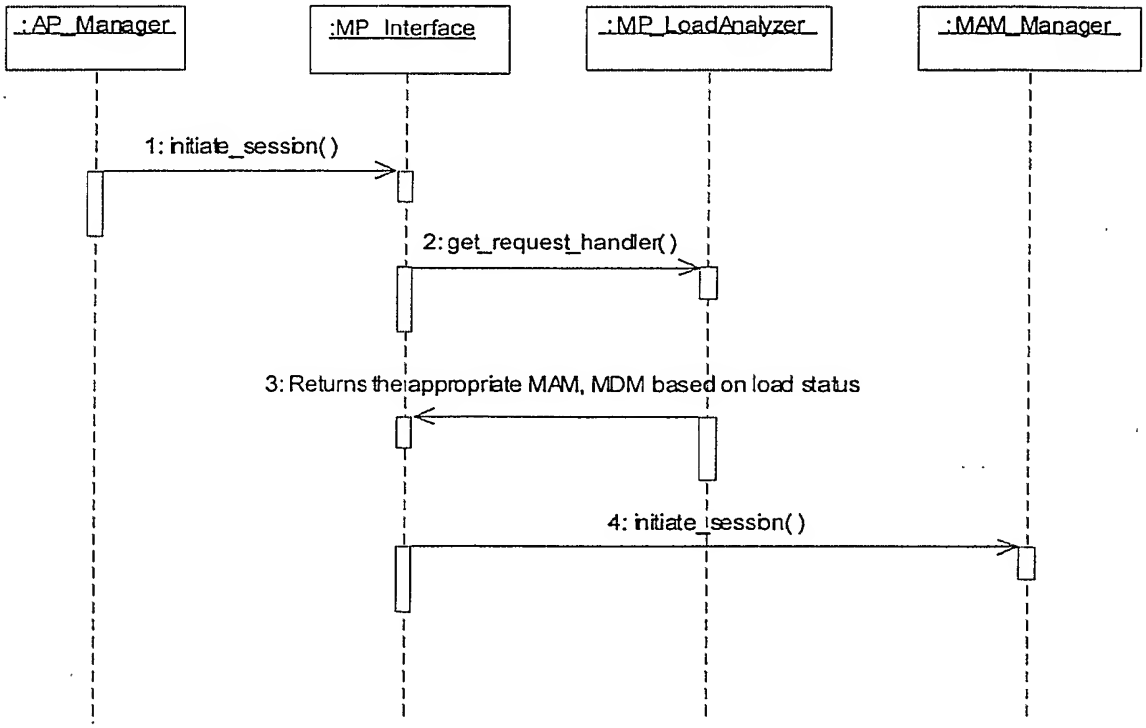
MEDIA PLANE USE CASE ANALYSIS

Use Case: Session Initiation Request to Media Plane Interface

The architectural significance of this use case lies in the fact that this use case follows from the Client Session initiation process in the application plane. The application plane manager invokes the session initiation request after parsing the XMT-O content and creating the session initiation parameters. The class **AP_Session_Initiation_Message** represents the session initiation request.

After receiving the session initiation request the Media Plane Interface with the collaboration of a load analyzer, decides the appropriate MAM and MDM, which can service the request. This process of determining the MAM and MDM is described in the next use case. The primary responsibility of the load analyzer is to provide load-balancing information.

The Media Plane Interface invokes the session initiation request on appropriate MAM after receiving the recommended MAM and MDM from the load analyzer.

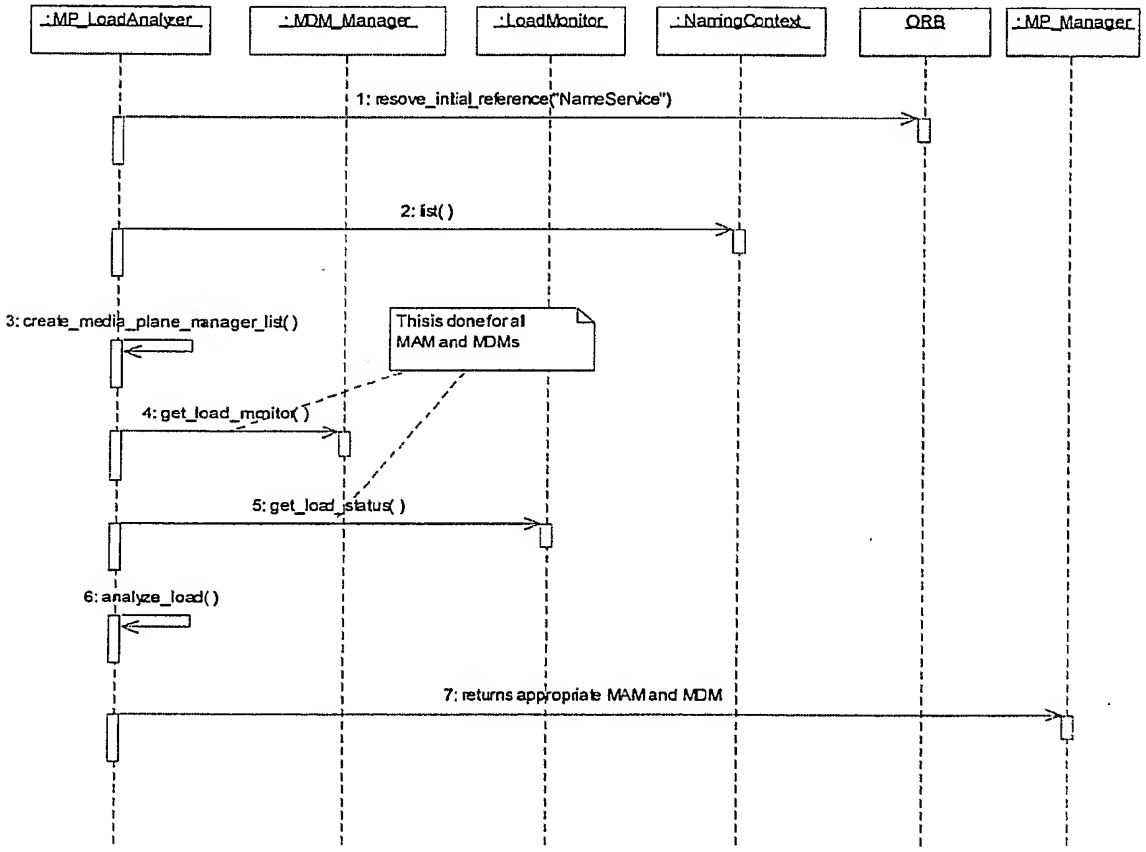


Use Case: Load Analysis

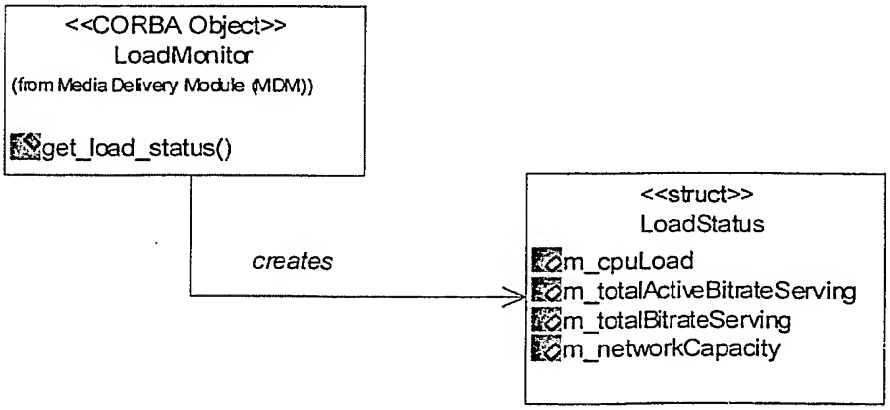
The load balancing strategy we are choosing is called adaptive per session load balancing strategy. In per session load balancing strategy, the client request will continue to be forwarded to the same module throughout the duration of the session. In the adaptive load balancing strategy, the load balancer utilizes run-time information such as current CPU load and network load on the line cards to select the appropriate line card to handle the request. We are choosing a load balancing strategy, which is a combination of these two strategies, called adaptive per session load balancing strategy. The per session load balancing is important because the every session has associated state information, so a request for one session will be totally out of context for another session because the relevant state information is not available. The adaptive strategy is important because the nature of the service we are providing is very interactive so based on user interaction the certain media stream will be stopped and new stream will be started. This leads to a situation where different load parameters are varying heavily based on user interaction. So the load balancer can take a more accurate decision if it takes into account the run time loading condition of the processor and network related resources.

In this use case the load analyzer is responsible to collect the load parameters from each media acquisition manager module and media delivery manager module, then it needs to analyze the load and find out the appropriate media acquisition manager module and media delivery manager module to handle the request. The loading condition analysis and determination algorithm will be very simple now. The line card with least CPU load, and least serving bandwidth (Aggregate bandwidth served, which is sum of all bit rates of streams served) will be chosen. The load analyzer

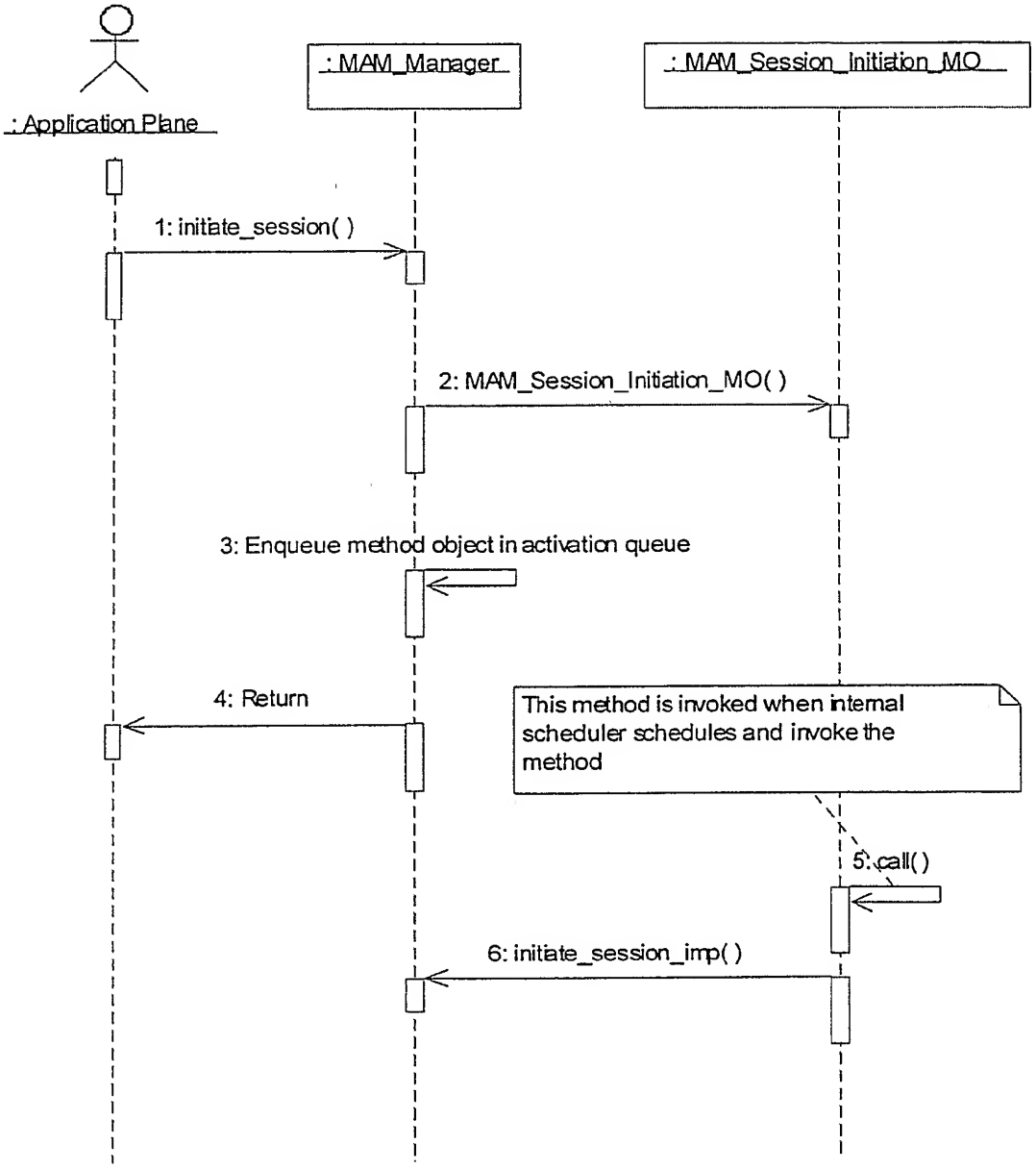
gets the Media Acquisition Module manager and Media Delivery Module manager from the CORBA naming service. The relevant managers will register them when the system boots up.



Following is the class diagram explains in more detail the LoadMonitor object and the LoadStatus data structure.



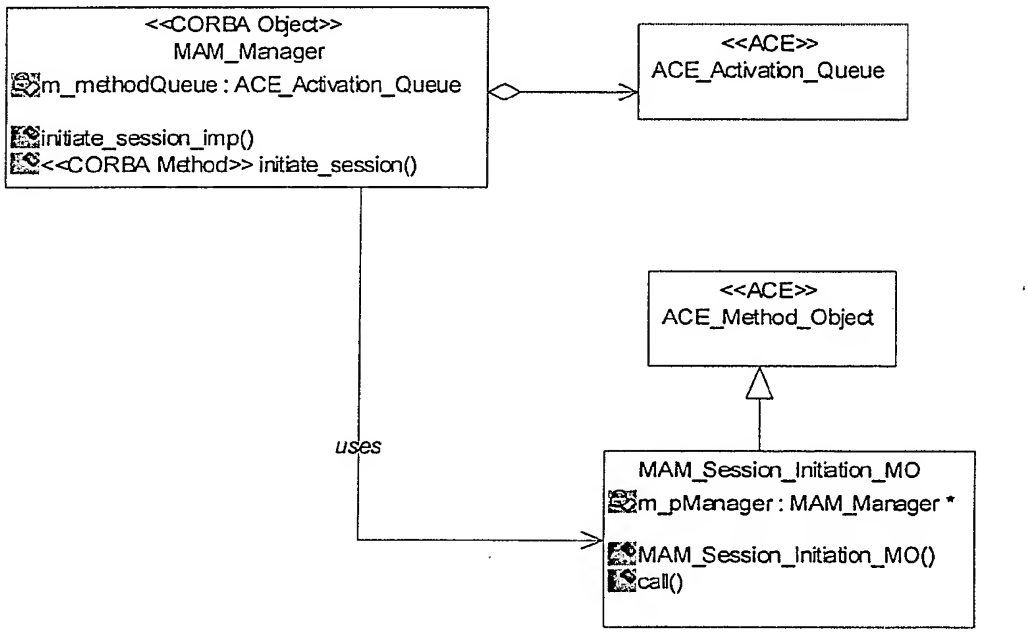
Use Case: Session Initiation Request To MAM



This is an architecturally significant use case because we have used the Active Object Pattern of ACE Framework. The Active Object Pattern is used to decouple the method execution from method invocation. When the Application Plane issues the method call initiate_session, this causes the corresponding method object MAM_Session_Initiation_MO to be instantiated and then enqueued on the activation queue. The initiate_session call returns immediately after that. The ACE_Activation_Queue is queue in which method objects are enqueued as they wait to be executed. Thus the activation queue contains all of the pending method invocations on it. The

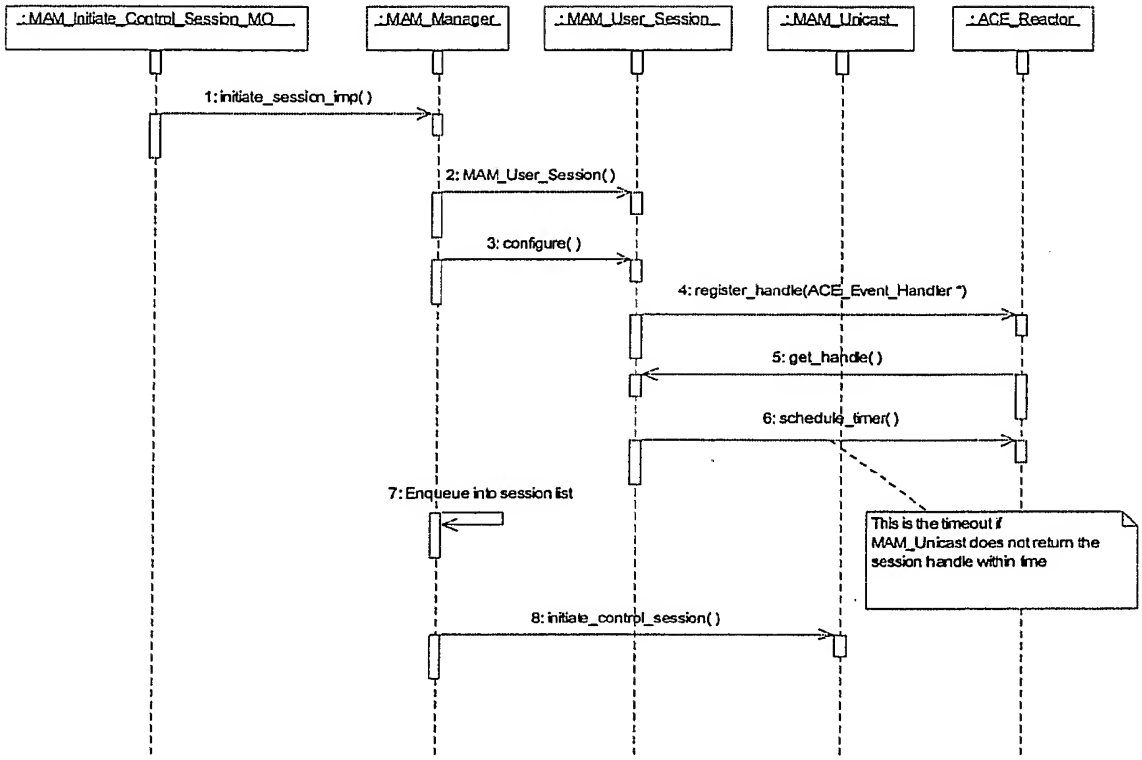
internal thread of MAM_Manager stays blocked, as it waits for any method object to be enqued on the activation queue. Once the method object is enqueued the thread dequeues the method object and invokes the call() method on it. The call() method of MAM_Session_Initiation_MO calls the initiate_session_imp() method that is the actual implementation of the method. This type of decoupling enables the Application Plane to invoke methods on Media Plane asynchronously and in a non-blocking way.

The corresponding static class relationships appear as follows.

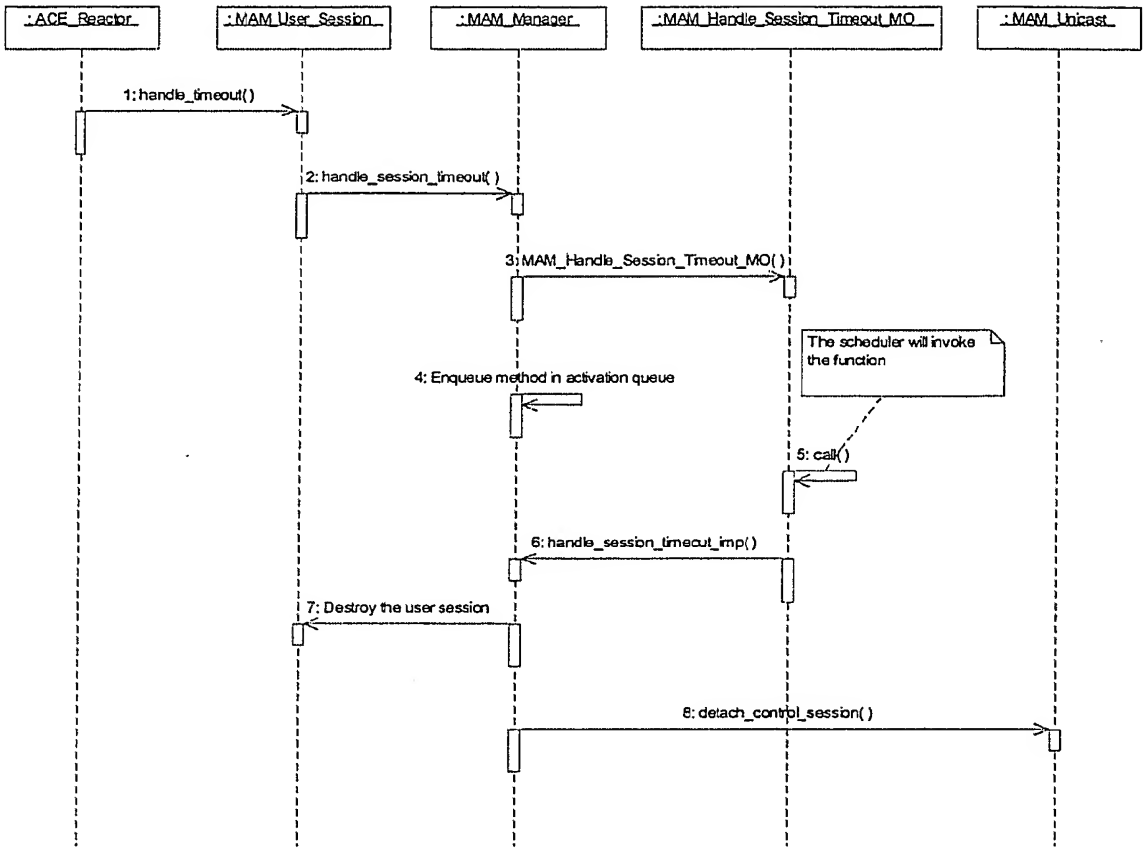


The MAM_Session_Initiation_MO retains a pointer to MAM_Manager, which is used to invoke the actual method. The MAM_Manager holds an activation queue that is used to enqueue the method objects.

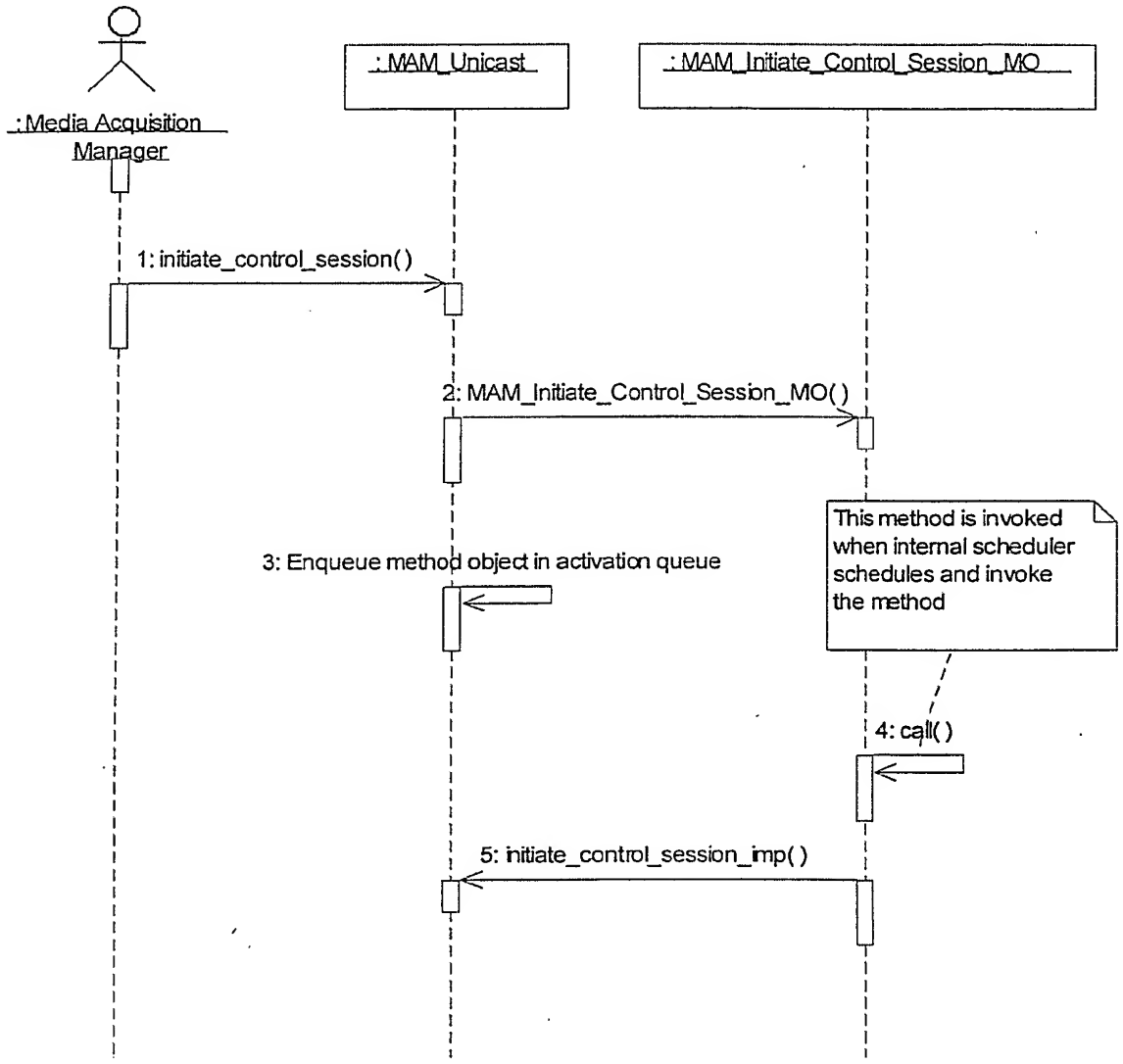
Use Case: User Session Creation



Use Case: User Session Timeout Handling

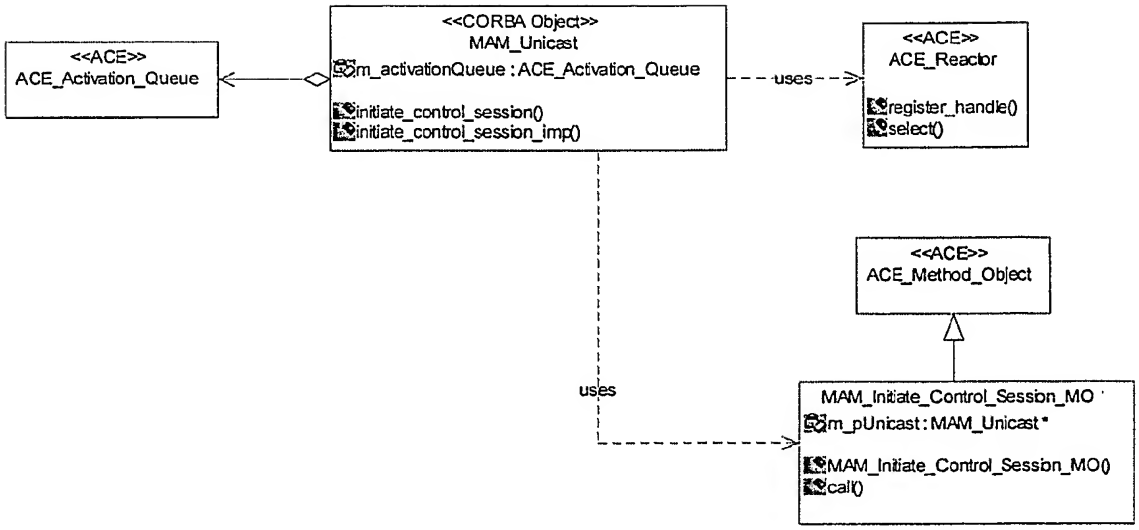


Use Case: Control Session Initiation Request To MAM_Unicast



This is an architecturally significant use case because we have used the Active Object Pattern of ACE Framework. The Active Object Pattern is discussed before. The initiate_control_session_imp() will initiate the connect call in use case RTSP Connection Establishment.

The static class diagram is described in the following figure.



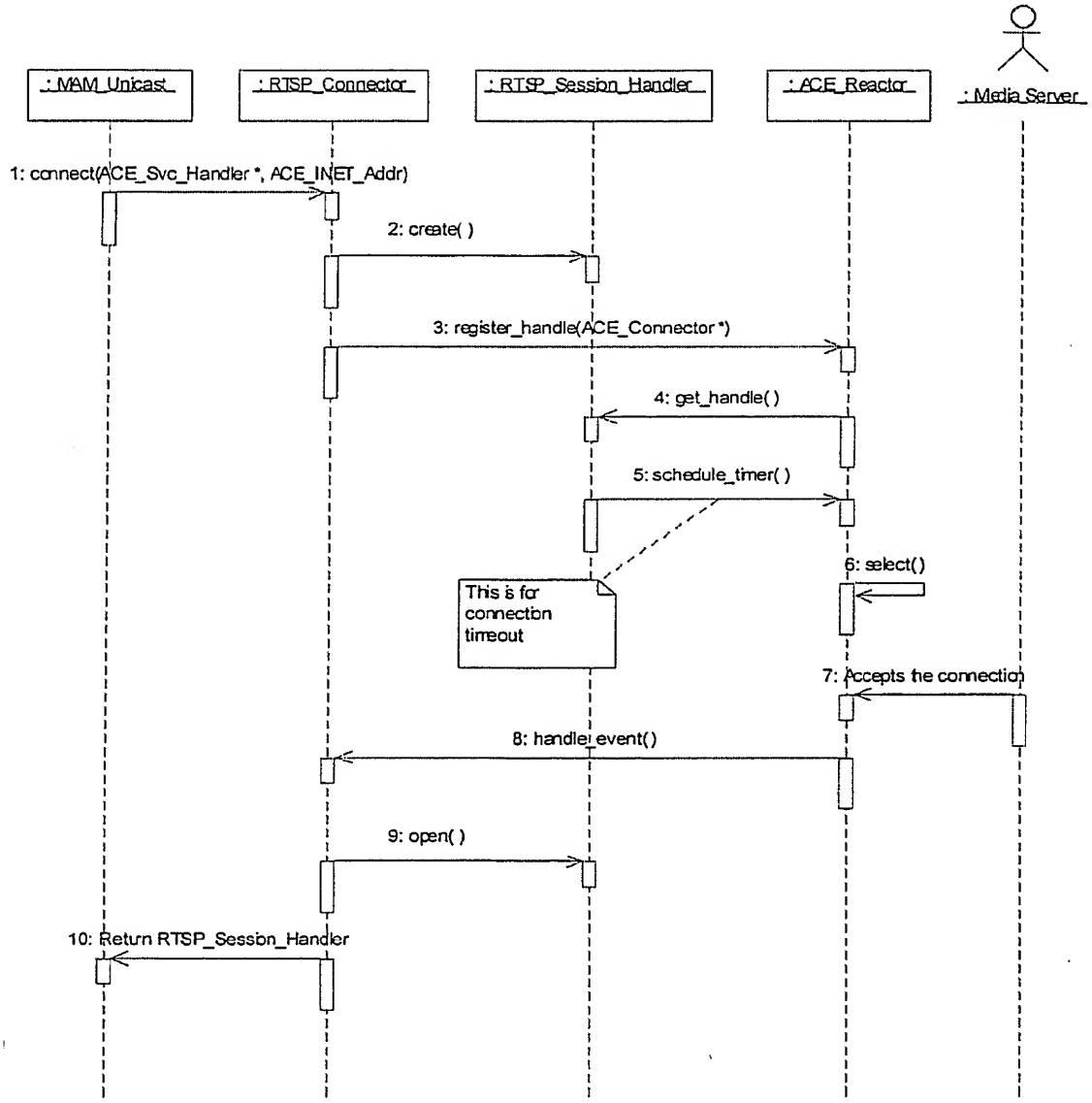
Use Case: RTSP Connection Establishment

This use case describes how RTSP connection is established by MAM. This use case is architecturally significant because we used the connector pattern of ACE to establish connection with the media servers. The use case is divided into following phases.

Connection Initiation Phase: This phase consists of steps 1 to step 7. The RTSP_Connector facilitates establishment of connection between RTSP_Session_Handler and its peer media server. The process of connection establishment happens asynchronously, which prevents the RTSP_Connector from being blocked. This facilitates other connect request processing while waiting for one or more pending connect request with Reactor. As described previously the Reactor provides a framework for OS level asynchronous event demultiplexing system calls with an extensible and portable callback driven object-oriented interface.

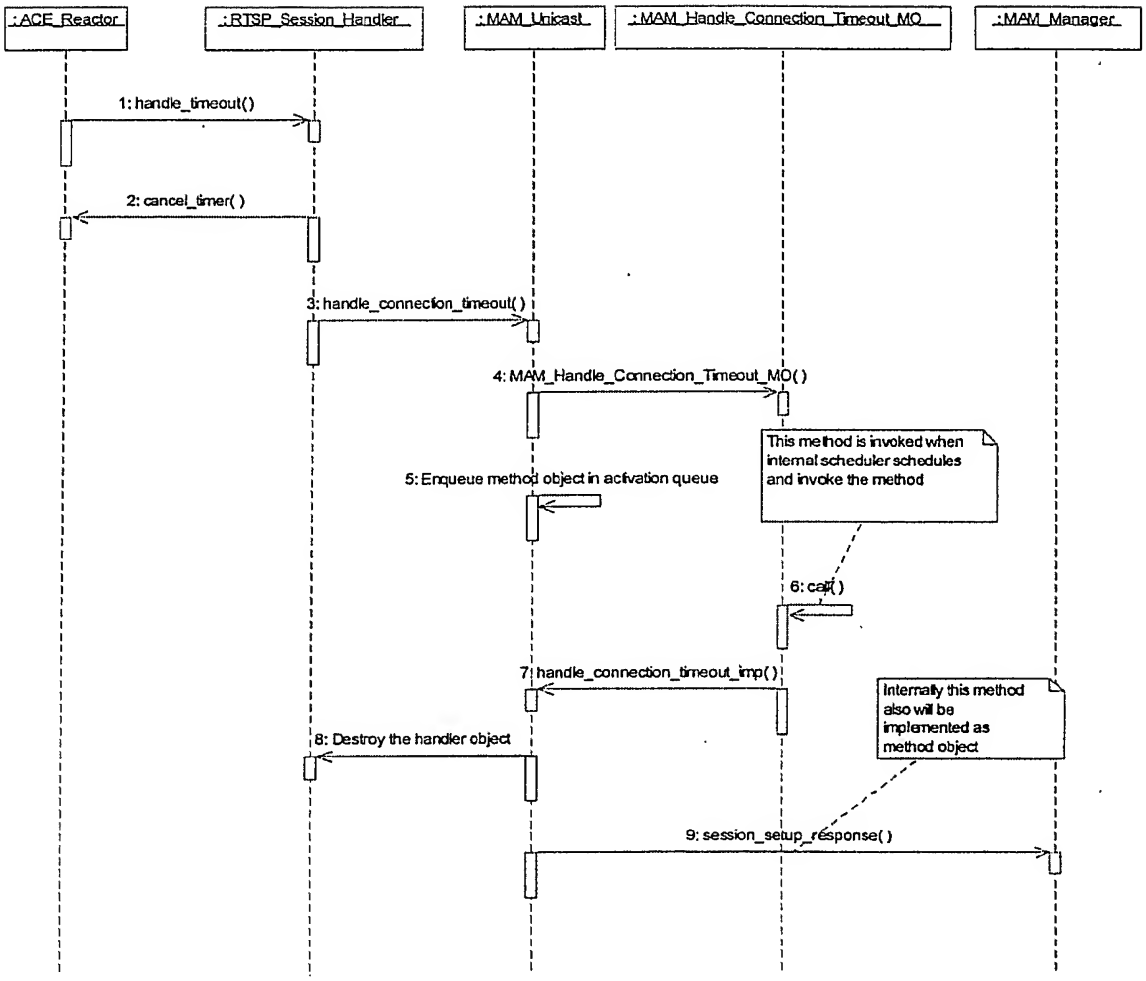
Service Initialization Phase: The Reactor notifies the RTSP_Connector when media server accepts the connection. The RTSP_Connector calls the initialization routine of RTSP_Session_Handler after it receives the notification from the Reactor. The RTSP_Session_Handler initializes all the internal data structures at this phase. The open() method is the initializing routine for RTSP_Session_Handler.

This use case also handles the situation where the connect request to the media server may fail and will never return. The RTSP_Session_Handler registers itself with the ACE_Reactor for a time out event. The ACE_Reactor will call back the RTSP_Session_Handler after the timer expires. The amount of time the RTSP_Session_Handler should wait needs to be decided later. The chain of events happens when the time out occurs are described in the use case RTSP_Connection_Timeout_Handling.



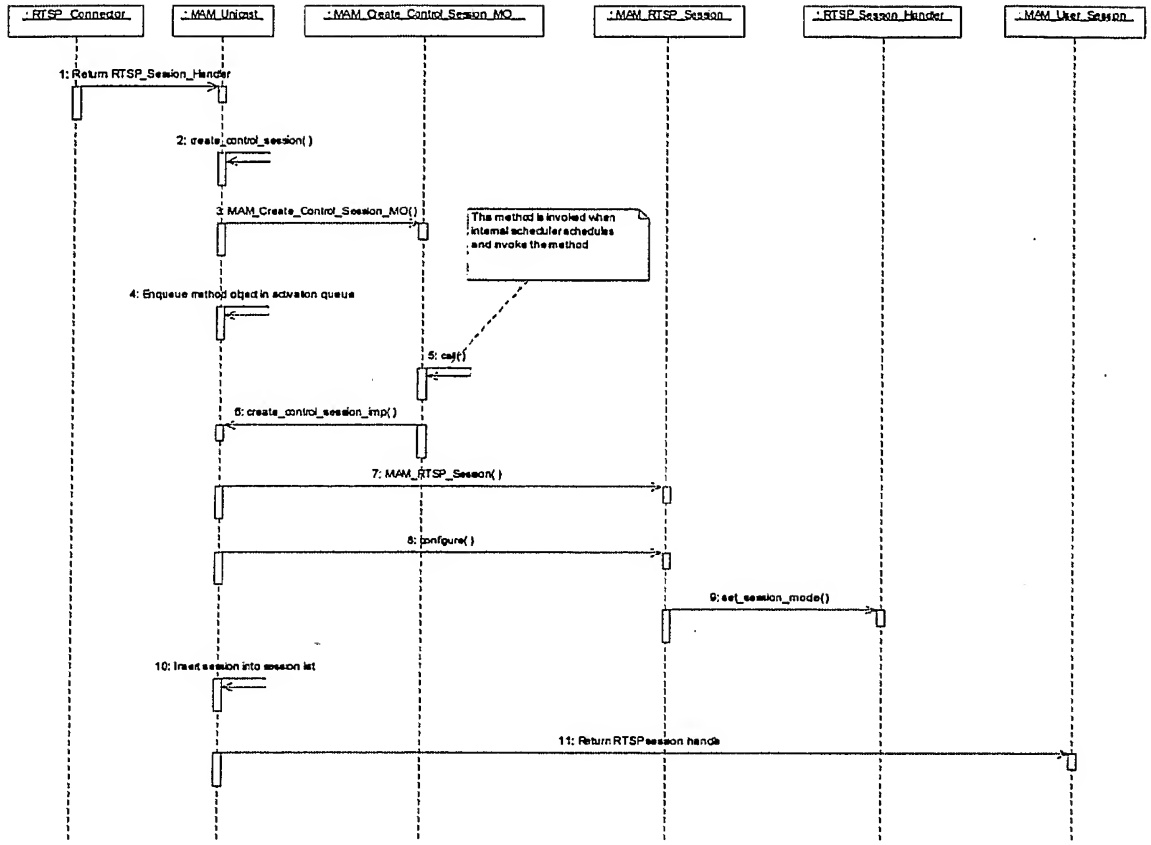
Use Case: RTSP Connection Timeout Handling

Though this use case is described in the context of RTSP connection establishment, the design philosophy can be used for any time out handling. This use case starts when the ACE_Reactor calls the handle_timeout() method of RTSP_Session_Handler. ACE_Reactor calls this method when the time out period of registered timer expires. The RTSP_Session_Handler passes this connection failure information to the MAM_Unicast. MAM_Unicast essentially destroys the RTSP_Session_Handler and sends the notification to MAM_Manager, who initially requested for the session initiation.



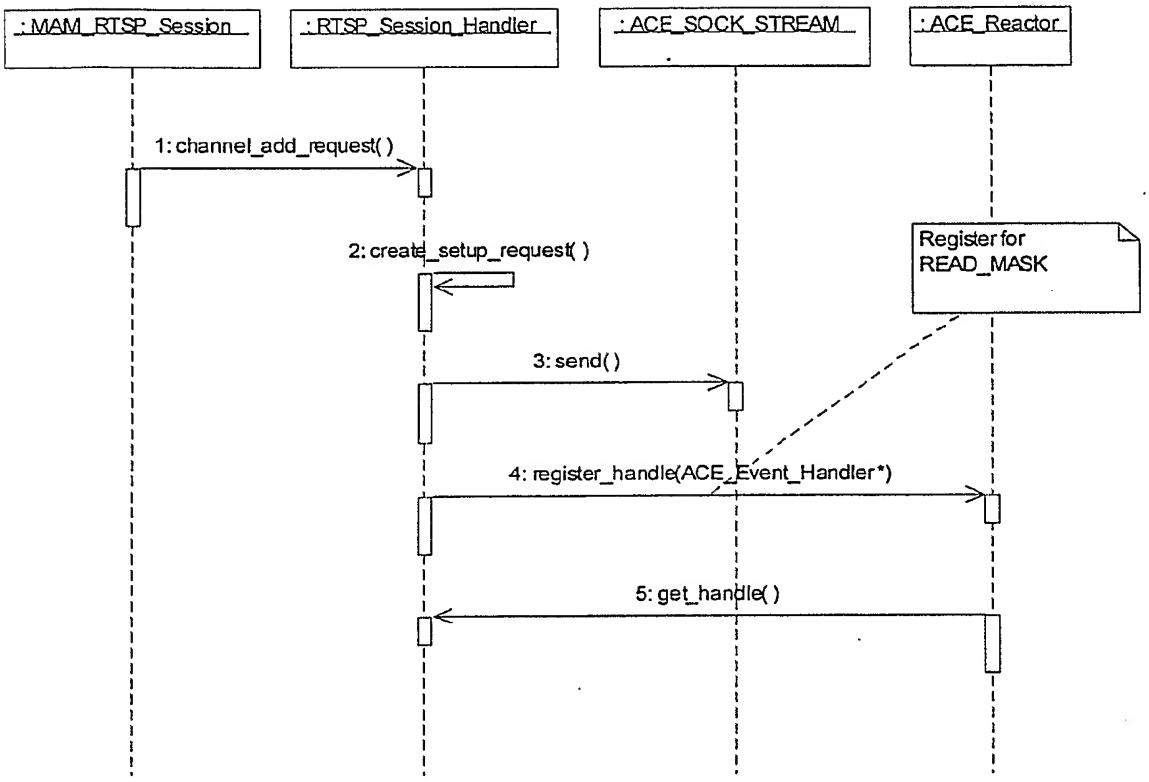
Use Case: RTSP Session Creation in MAM

The precondition for this use case is that the connection with the media server has been established successfully. This use case describes the RTSP session creation process in details.



Use Case: Channel Add Request

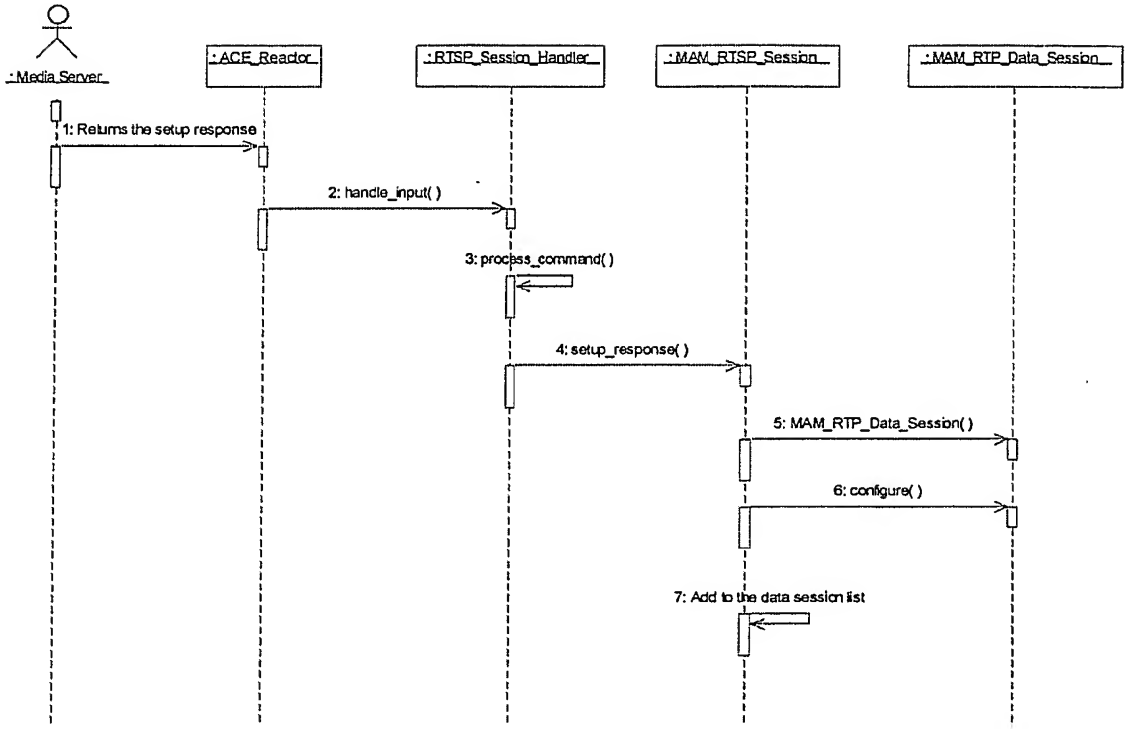
An RTSP session gets the stream information from the application plane. The stream information describes the server location, stream id etc. The RTSP session requests the RTSP session handler to send a channel add request. The create setup request function creates a setup request in a buffer. The RTSP session handler generates a unique port number for receiving RTP data as well as RTCP data (If the UDP ports are used to receive data in a non-tunnel mode) and attaches the port number with the request.



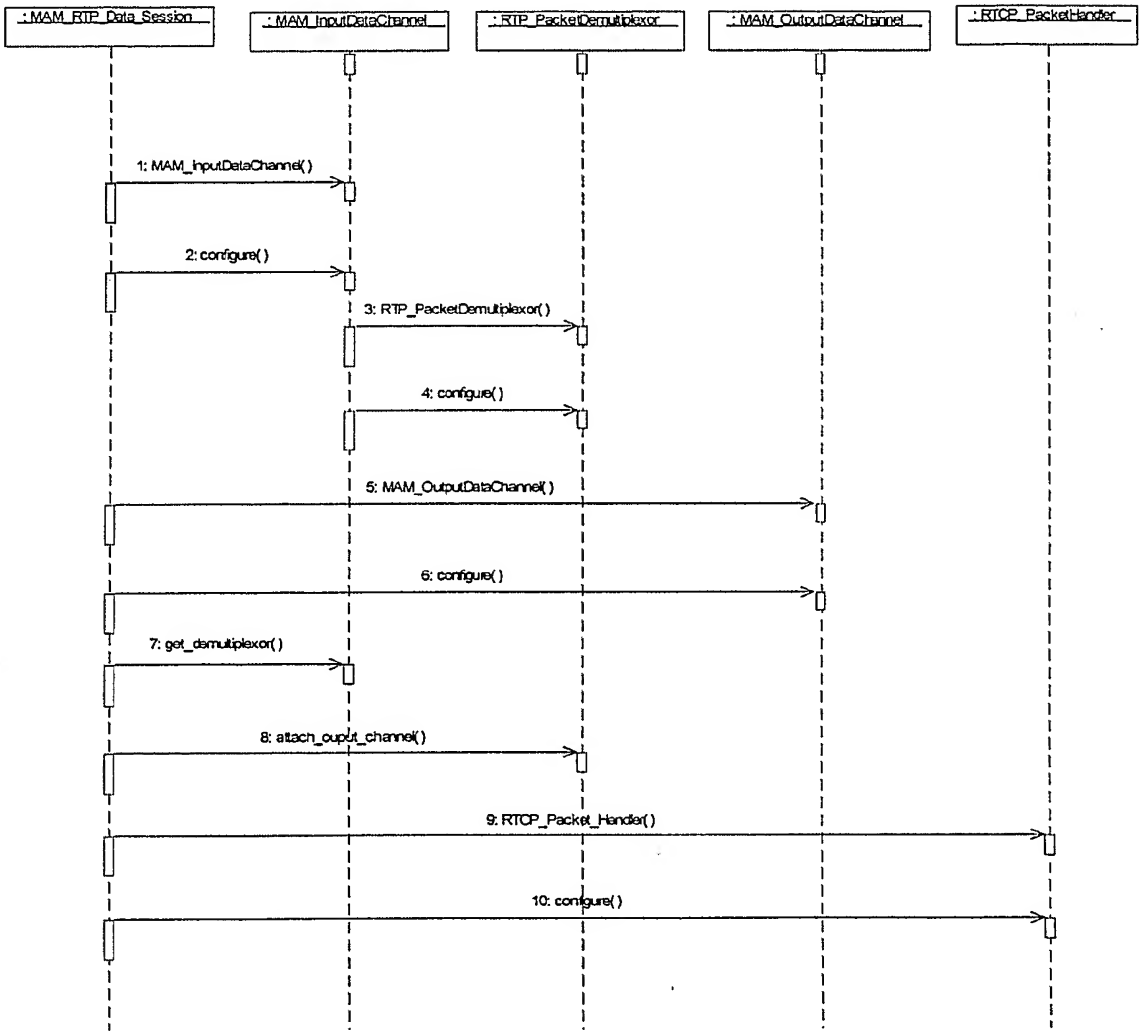
Use Case: RTP Data Session Creation

According to the previous use case, the RTSP session handler waits in the ACE reactor for a read event. When media server sends the setup response, the ACE reactor calls the handle_input() callback of RTSP session handler. The RTSP session handler then parses the command and constructs a RTSP response command. Then the RTSP session handler invokes the setup_response() method of RTSP session. This method is primarily responsible for instantiation and configuration of RTP session.

Details of RTP session configuration is described in a different use case. The RTSP session handler sets its state to WAITING_FOR_SETUP_RESPONSE after sending the setup command to the media server. This state information is used to process the response from the media server. If the RTSP session handler is in a different state than the media server response, then it needs to raise the exception.


:Media Server
:ACE_Reactor
:RTSP_Session_Handler
:MAM_RTSP_Session
:MAM_RTP_Data_Session
1: Returns the setup response
2: handle_input( )
3: process_command( )
4: setup_response( )
5: MAM_RTP_Data_Session( )
6: configure( )
7: Add to the data session list

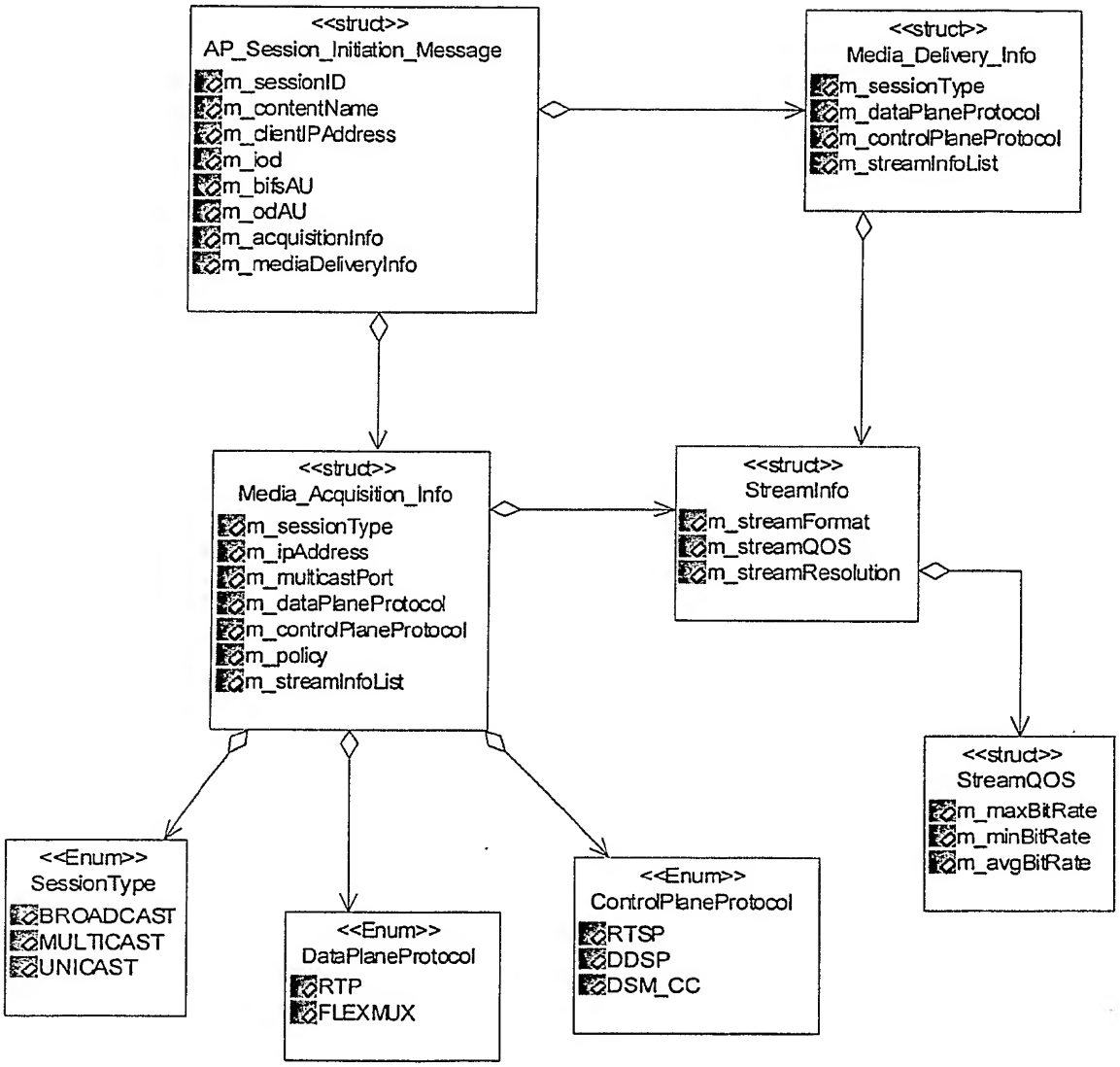
Use Case: RTP Session Initialization and Configuration



MAM LOGICAL VIEW

The use case view captures the dynamic nature of the system, but the logical view mostly captures the static aspect of the system. This view defines the architecturally significant classes and their relationships with each other. A class diagram captures a related set of classes and their relationship. The class diagram will be architecturally significant if it illustrates any design pattern, important framework or important infrastructure of the system.

App Session Initiation Message



The application plane manager constructs the AP_Session_Initiation message after parsing the XMT content. This message is sent as argument for session initiation request. The sessionID is a unique id in the context of client sessions in the application plane, which is generated by application plane manager. Based on the information contained in media acquisition info, the MAM module configures itself and based on the information contained in media delivery info, the MDM module configures itself.

MAM Overall Session Relationship

This diagram illustrates the high level classes and their relationships.

MAM_Manager: The responsibility of this class is to

- Provide an abstract interface to the outside world for Media Acquisition Module.
- The outside modules in application plane, management plane and network plane communicate with this class to request for information, provide information, provide and receive control messages.

MAM_User_Session: This class represents a unique user session. A user session can be created in response to a user request for media service delivery.

- The user session aggregates other media sessions and associated streams. The MAM_Manager is responsible to create and destroy the user sessions. MAM_Manager also is responsible to contain the list of user sessions.

MAM_Unicast: This class is responsible for

- Providing an abstract interface to the rest of the Media Acquisition Module for communicating with the unicast MAM module. The unicast MAM module's responsibility is to establish unicast one-to-one session with the media servers or other network devices and fetch media. The unicast MAM is similar to unicast client DMIF of MPEG-4 framework.
- The MAM_Unicast class provides a network independent application layer interface for creating, destroying unicast sessions.
- Also it provides interface for collecting usage information from unicast sessions. Through callback interface it can send fault and error information to other modules.

MAM_Control_Session: This class represents the abstract unicast control session. This class is an abstract base class from which a control protocol specific concrete will be derived. The responsibility of the control session is to

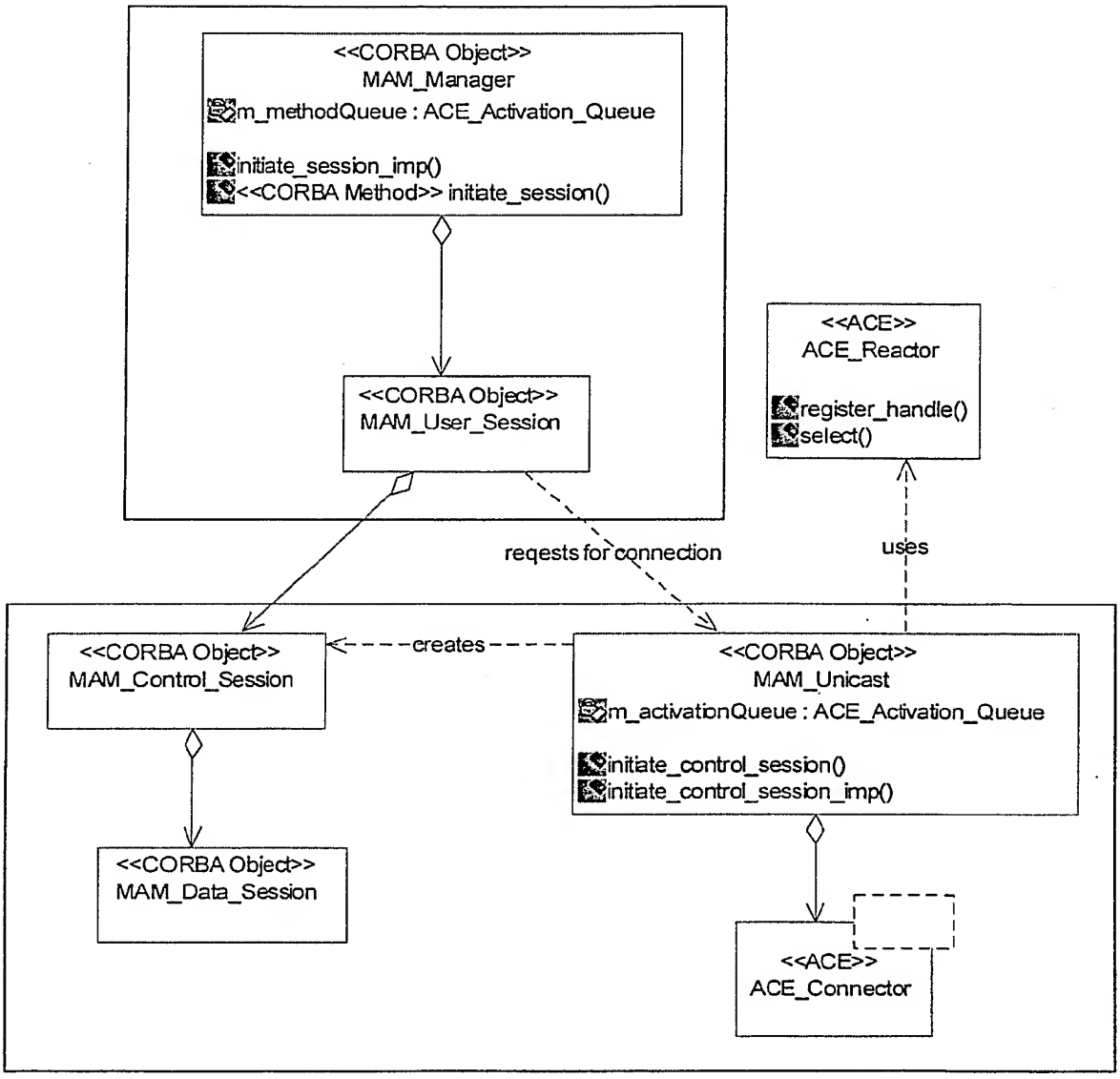
- Establish connection with media servers to establish unicast session
- Send and receive control messages to and from the media servers to initiate data sessions to receive media, to control streams etc.
- Create and aggregate data sessions which receives media streams

MAM_Data_Session: This class represents a unicast media data session. This class is an abstract base class from which a media protocol specific concrete class will be derived. The primary responsibility of data sessions is to

- Provide a network interface (SOCKET) to receive media streams from media servers.
- Demultiplexes the transport packets and extracts media payload.
- Aggregates fragmented media payloads into one AU.
- Processes media AU as necessary and transmit it to Media Delivery Module.
- Collects network traffic management parameters such as jitter, latency, packet loss etc through follow control protocol and provide it to traffic shaper if necessary.

The MAM_Manager and MAM_User_Session reside into a CORBA server and MAM_Unicast and associates objects resides into a separate CORBA server. This separation was done because of the following reasons

- The Unicast Module, Broadcast Module, Multicast Module can reside in different line cards.
- The MAM_Manager can aggregate different types of session in one user session from different line cards.



The MAM Media Control Session Relationship

This class diagram captures relationships between different control sessions in Unicast Media Acquisition Module. The responsibility of control session is described earlier. This diagram provides relationship between abstract control session class and different concrete control session classes.

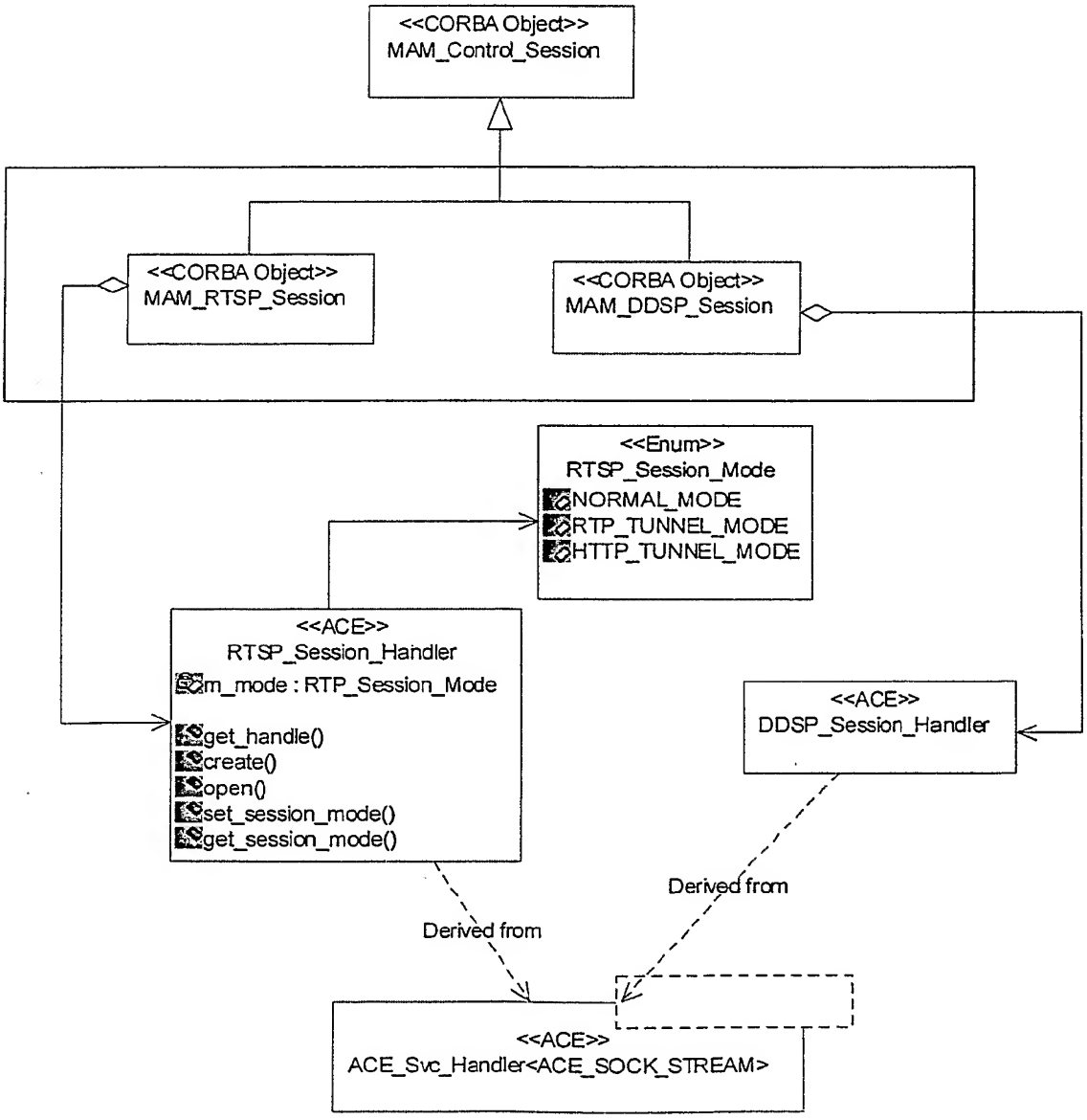
MAM_RTSP_Session: This class is a concrete control session class, derived form the abstract base MAM_Control_Session class. Following are the responsibility of this class

- Provides a protocol independent interface to rest of the IMCE modules to send and receive control messages.
- It translates the protocol independent messages to RTSP messages and translates RTSP messages to a protocol independent message.

- It sends and receives RTSP based messages to and from media servers.

MAM_RTSP_Session_Handler: MAM_RTSP_Session_Handler class encapsulates Object of this class. Following are the responsibilities of this class

- It encapsulates a network interface (SOCKET) through which RTSP messages are sent to the media servers through the network and RTSP messages can be received from the network.
- It operates in two different modes. In NORMAL_MODE it just sends and receives RTSP messages. In RTP_TUNNEL_MODE in addition to the functionality described, it sends multiplexed RTP data streams through the network interface.



The MAM Data Session Relationships

This class diagram represents the data plane architecture of Media Acquisition Module. This class diagram captures major data plane objects and their static relationship.

MAM_RTP_Data_Session: This class is derived from abstract base data session class. Following are the responsibilities of this class.

- This class is responsible establish network connection with the peer media server and receive media stream in RTP packetized format through network interface.
- This class is responsible to parse RTP packets and extract fragmented media payloads out.
- This class is responsible for forming an integrated media payload by concatenating fragmented media payloads.
- This class is responsible for transmitting media streams to Media Delivery Module through mutually established network interface.
- This class also is responsible for measuring the latency, jitter and other provisioning network congestion parameters through RTCP flow control protocol.

MAM_Input_Data_Channel: This class has the following responsibilities

- The main functionality is to encapsulate the network interface (SOCKET) and media payload extraction from transport packets.
- The input data channel hosts a series of processing filters in a pipelined fashion, and packetized media stream flows through the pipeline and get processed at each stage.
- This input channel interfaces with one or more output data channel to transmit media to Media Delivery Module.

MAM_Output_Data_Channel: The class has the following responsibilities

- The main functionality is to encapsulate the network interface (SOCKET) through which the media stream is delivered to Media Delivery Module.
- This class like input data channel hosts a series of processing filters in a pipe lined fashion, and packetized media stream flows through the pipeline and get processed at each stage.
- The major functionalities provided by the processing filters are media payload reassembly into an AU, decryption of AU, transcoding of AU if necessary.

RTP_Packet_Demultiplexer: The class has the following responsibilities

- The main responsibility of this class is to read transport packets from the network interface (SOCKET) and extracts media payload out of it. This class specifically reads and processes the RTP transport packets.
- In case where more that one media stream is multiplexed, this class provides a demultiplexing interface to separate stream specific transport packets.

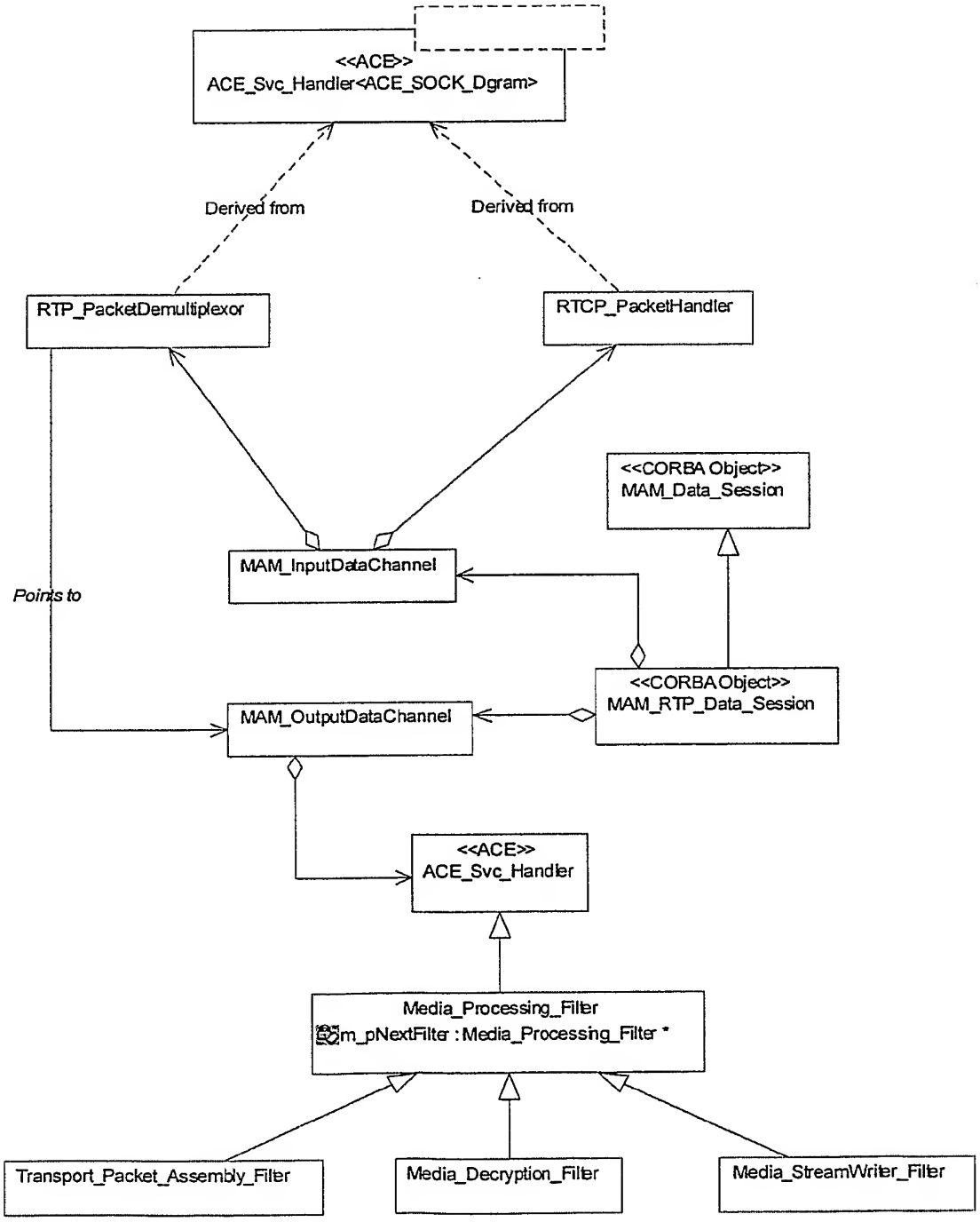
RTCP_Packet_Handler: The class has the following responsibilities

- The main responsibility of this class is to encapsulate the RTCP flow control protocol, which means generating RTCP related parameters, send RTCP feed back to peer etc.
- Also this class is responsible for maintaining the RTCP related states so that the upper layer objects can query it.

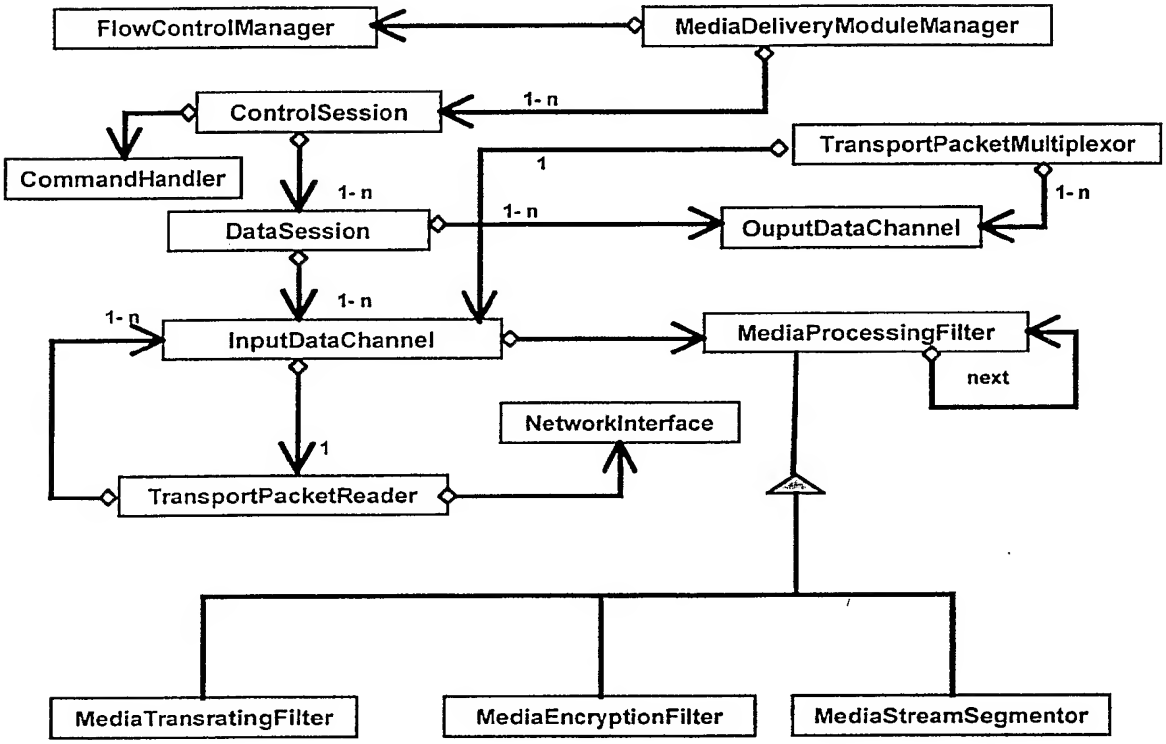
Media_Processing_Filter: The class has the following responsibilities

- This class serves as an abstract base class for media processing filters. The media processing filters are processing blocks, which take media packets as input then processes the media and sends the processed media packets as input to another Media Processing Filters. Example of concrete media processing filters are decryption filter, transcoding

filter etc.


<<ACE>>
ACE_Svc_Handler<ACE_SOCK_Dgram>
Derived from
Derived from
RTP_PacketDemultiplexor
RTCP_PacketHandler
<<CORBA Object>>
MAM_Data_Session
MAM_InputDataChannel
Points to
<<CORBA Object>>
MAM_RTP_Data_Session
MAM_OutputDataChannel
<<ACE>>
ACE_Svc_Handler
Media_Processing_Filter
m_pNextFilter : Media_Processing_Filter *
Transport_Packet_Assembly_Filter
Media_Decryption_Filter
Media_StreamWriter_Filter

MEDIA DELIVERY MODULE ARCHITECTURE



GLOSSARY OF TERMS

Italicized terms in glossary definitions are defined elsewhere in the glossary.

Access Network: the network connecting an *IMCE* and its clients. Cf. *Provisioning Network.*

Access Unit (AU): a decodable media payload containing sufficient contiguous media data to allow further processing. For visual media an Access Unit represents one encoded frame. If an Access Unit is larger than the maximum transfer unit (MTU), it may be segmented into smaller pieces for transmission and then reassembled by the receiver. If an Access Unit is much smaller than the MTU, then multiple Access Units can be multiplexed together in one transport packet.

aggregation: combining media streams into a single stream.

application-aware: a device or module that can adapt to the current state of applications in the network. For example, if the user reduces the size of a video window, an application-aware device can process the media at a lower resolution.

Application Plane: the *IMCE* component responsible for connecting it with MPEG-4 servers, content packagers, and other IMCEs. The Application Plane receives and parses MPEG-4 Scene information in XMT-O form, associates it with, or initiates, a *client session*, extracts *network provisioning* information, and communicates this to the *Media Plane* via the *Media Plane Interface.*

Asynchronous Digital Subscriber Line (ADSL): a modem technology that converts existing twisted-pair telephone lines into access paths for multimedia and other high-speed data communications.

blade-based: *line card*-based.

Content Delivery Network (CDN): An edge provisioning network providing the fastest possible access to high bandwidth content. Examples include Akamai.

control sessions: A *network session* that manages control commands for stream control and session management and aggregates and manages *data sessions*.

data sessions: A *network session* that aggregates channels through which media flows to a client.

dejittering: adjusting the spacing between frames to eliminate *jitter*.

Intelligent Media Application Platform (IMAP): A companion product to the *IMCE* that generates personalized rich media applications based on an MPEG-4 Scene graph and system framework and dynamically configures IMCEs based on the targeted application's acquisition and delivery requirements. An IMAP and its associated IMCE form a *media application delivery platform.*

Intelligent Media Content Exchange (IMCE): a software and hardware product that gets media data from media servers via the *Provisioning Network*, aggregates the content, processes it, and delivers it to media clients via the *Access Network.*

Intelligent Media Delivery Platform (IMDP): an internal development name for the *IMCE.*

IMDP Payload Packager: a software module that combines BIFS and OD commands, XMT-O layout and graphics information, and client session and application identification information into a well-formed, valid XML document suitable for inclusion in an HTTP POST command.

jitter: variation in the delay between packets.

latency: the total delay experienced in response to a query, including delay from transmission devices, the network, and servers.

line card: single board computer that plugs into a rack with a backplane.

load analyzer: a software or hardware module that provides load-balancing information for stream processing.

macro block: a 16-by-16 pixel segment of a video frame used for motion vector analysis, computation, and estimation.

Management Plane: the *IMCE* component responsible for administering, managing, and configuring the IMCE.

media application: a media-centric application that typically involves streaming media.

media application delivery platform: a system consisting of both an *IMAP* and an *IMCE*.

Media Acquisition Module (MAM): the *IMCE* component responsible for aggregating media data from media servers or peer IMCEs over heterogeneous networks and delivering the media data to a *Media Processing Module* or *Media Delivery Module*. The MAM removes network jitter before delivering media data.

Media Delivery Module (MDM): the *IMCE* component responsible for receiving media data from a *Media Processing Module* and delivering the media data to multimedia clients or a peer *IMCE*. The MDM provides edge media processing. The MDM configures the data channel for each client, establishing a session with either a specific client or a multicast data port.

media network element: a network element optimized for media traffic.

Media Plane: the *IMCE* component responsible for stream acquisition, stream processing and stream delivery, which are provided by, respectively, the *Media Acquisition Module*, the *Media Processing Module*, and the *Media Delivery Module*.

Media Plane Interface (MPI): the programming interface for the *Application Plane*, *Management Plane*, and *Network Plane* to send messages to and receive messages from the *Media Plane*.

Media Processing Module (MPM): the *IMCE* component responsible for processing *access units* from a *Media Acquisition Module* and delivering processed access units to a *Media Delivery Module*. Possible MPM processing includes transcoding, decryption, and encryption.

network edge: the point where the network joins access technologies such as cable and wireless connections that reach local clients with the high-speed network core. For networks with an *IMCE*, the edge of the network is the boundary between the *Provisioning Network* and the *Access Network*.

network element: a processor on a network through which network traffic flows.

Network Plane: the *IMCE* component responsible for interfacing with other intelligent network elements such as routers and content routers, configuring the network environment for quality of service (QoS) provisioning, and maintaining routing tables.

network provisioning: providing hardware and software network resources to enable effective network processing and delivery of content and control data. Network resources being provisioned include those for stream processing, bandwidth, quality of service, and multiplexing.

network session - A session for communicating either data (*data session*) or control information (*control session*) with a client. Network sessions are maintained by a MAM and its associated MDM.

object descriptor (OD): an MPEG-4 object that identifies and names elementary streams so that they can be referred to in a scene description and be attached to individual objects. An OD is transmitted in its own elementary stream. Object descriptors are separate from the scene description itself, thus simplifying editing and remultiplexing of MPEG-4 content.

policy decision: a set of precise rules that match the needs of the application to the resources available from the network. These rules specify such things as the amount of bandwidth to be allocated to a specific application, relative priority, or time of day when policy needs may be enforced.

policy decision point (PDP): the entity that decides which policy to apply to a particular connection.

process filter graph: a sequence of processing filters connected with each other via ports, each of which filter performs a processing step such as real-time media analysis.

Provisioning Network: the portion of the network providing inputs to the *IMCE*, including inputs from media, scene, and Internet servers. Cf. *Access Network.*

service level agreement (SLA): a contract between network service provider and customer that guarantees a level of service, including such network performance parameters as packet loss, *throughput*, and *latency*.

session: A persistent stateful connection between two entities. The IMCE creates a unique application session and associated user session for each user to whom a media application is served. The user session aggregates different *network sessions*.

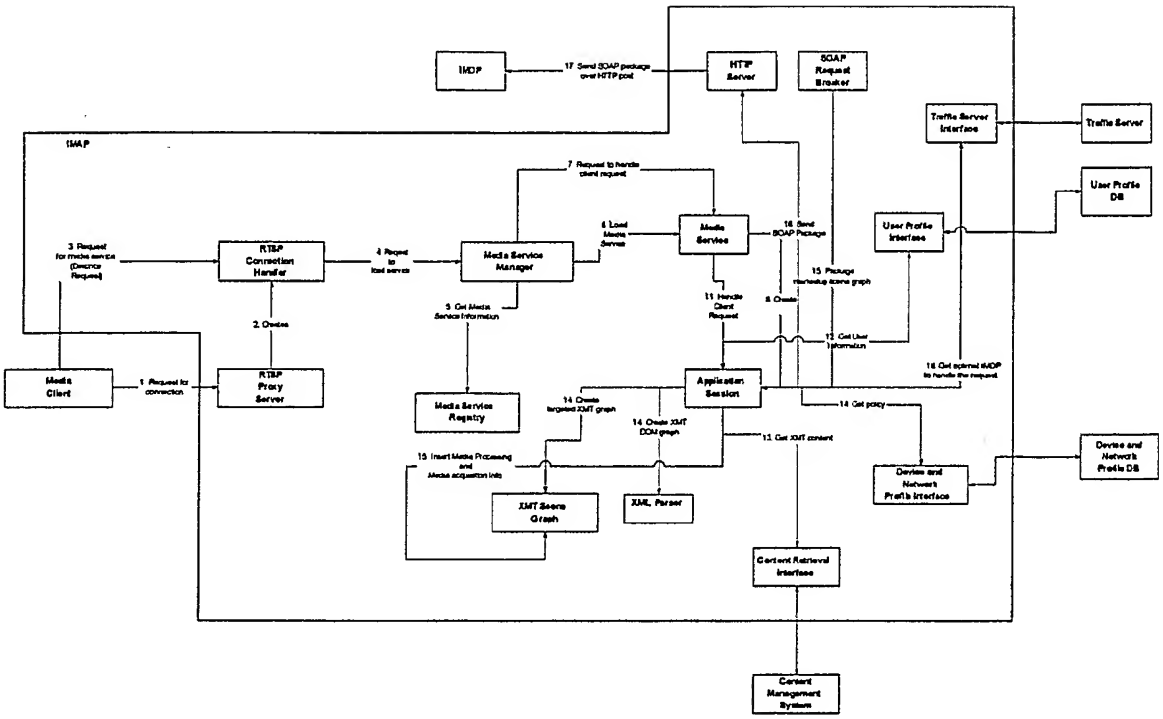
Synch Layer (SL): an MPEG-4 layer responsible for timing and synchronization information, such as time stamps and clock references. SL information allows a client to determine which portions of different streams are to be presented at the same time. In the *IMCE* the SL is responsible, based on timing and synchronization information, for breaking up media *Access Units* in the server and reconstructing and scheduling them in the client.

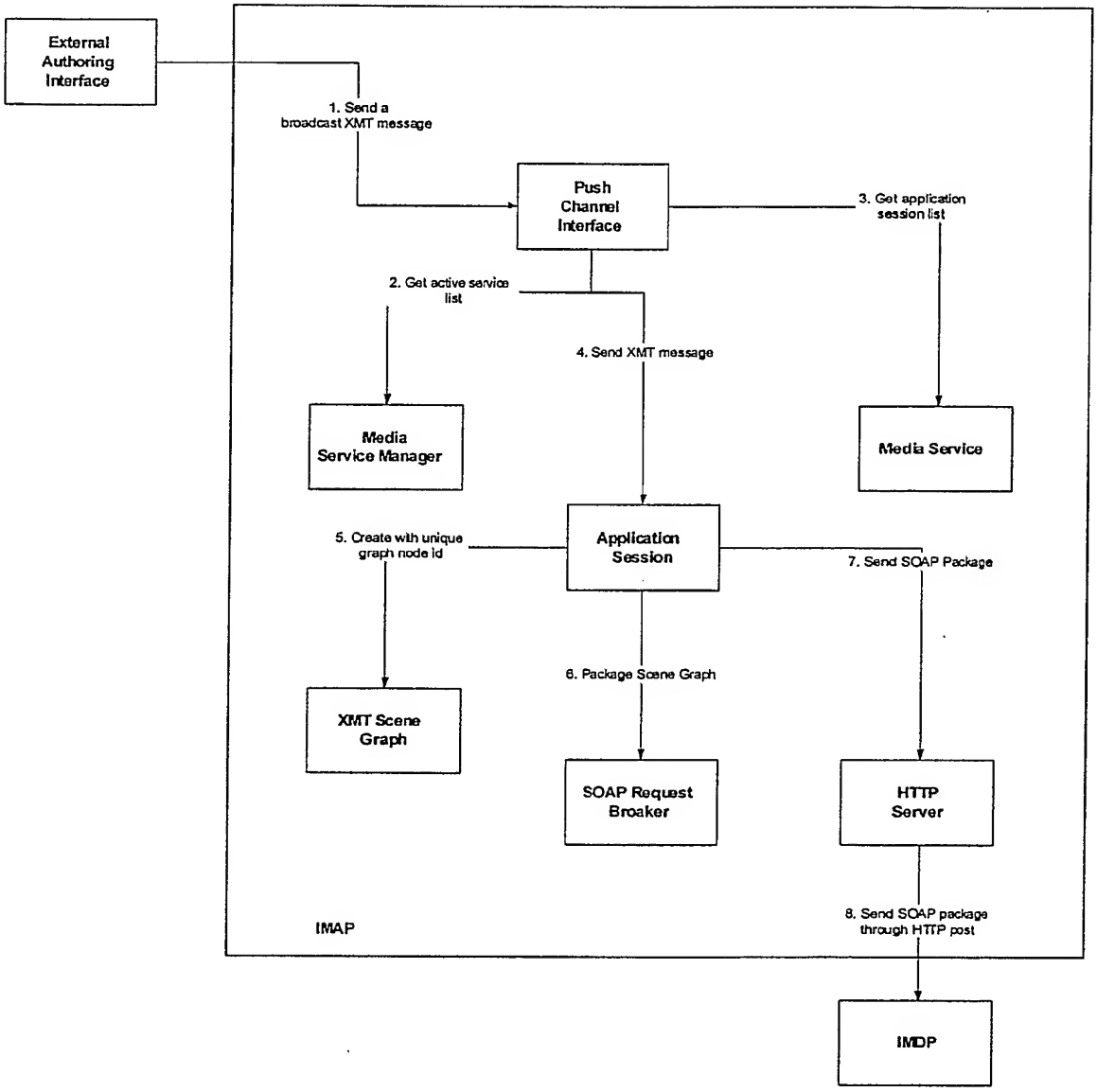
stream flaw: any error or defect in the media stream, including but not limited to bit errors, frame dropouts, timing errors, and flaws in encoding.

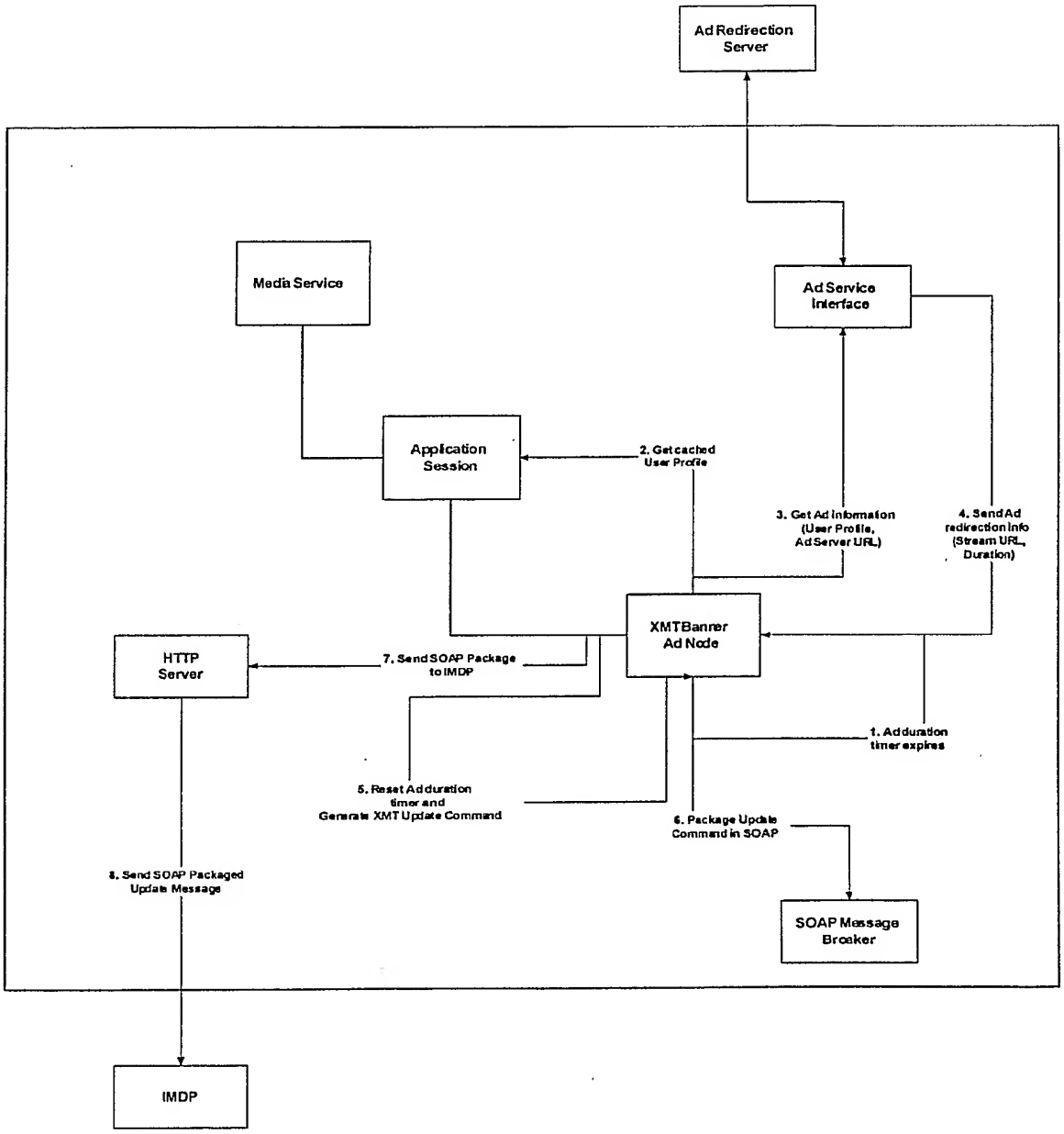
throughput: the average network traffic over a given time interval, expressed in bits per second (bps).

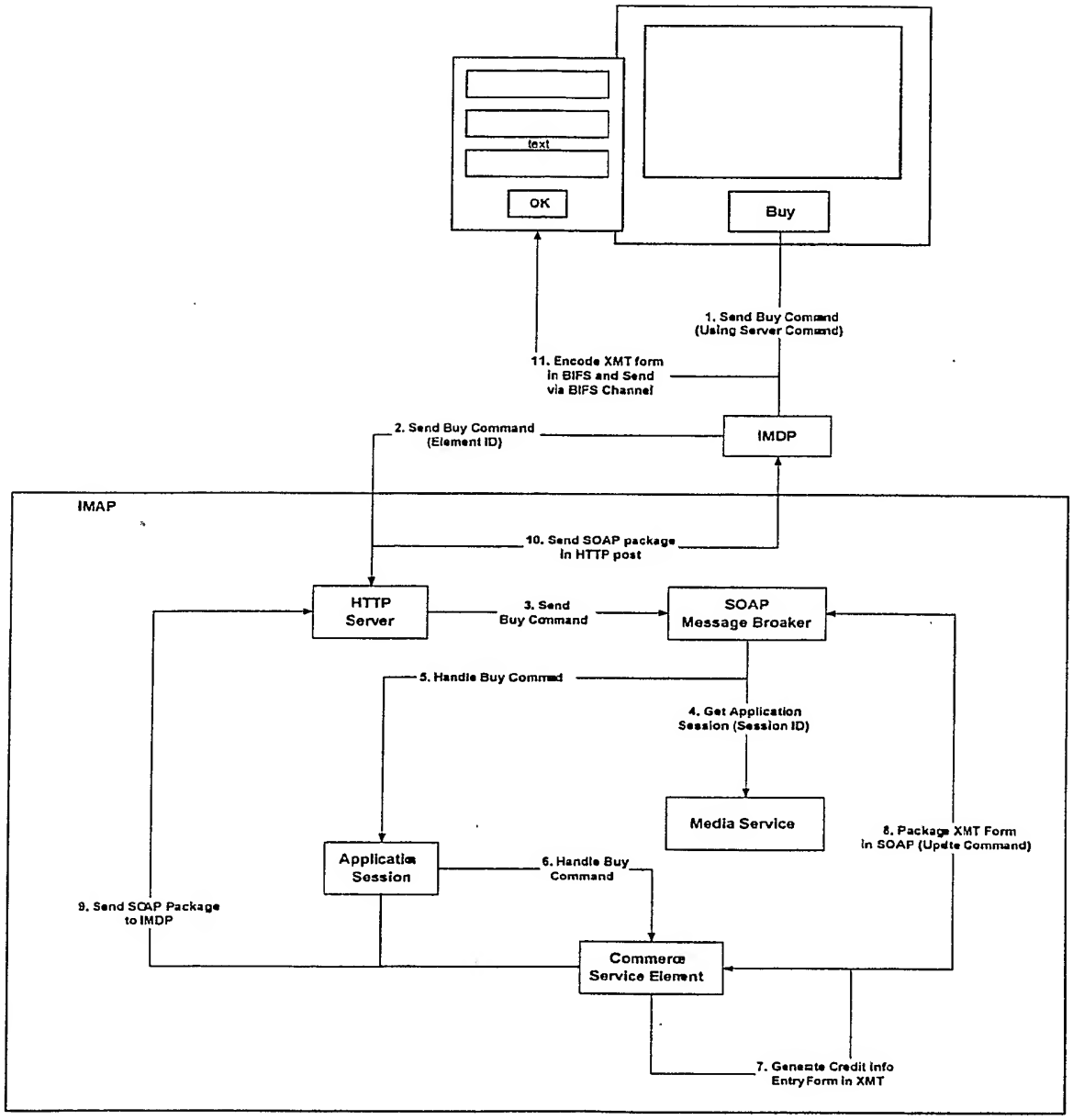
transrate: to change the bit rate of media data.

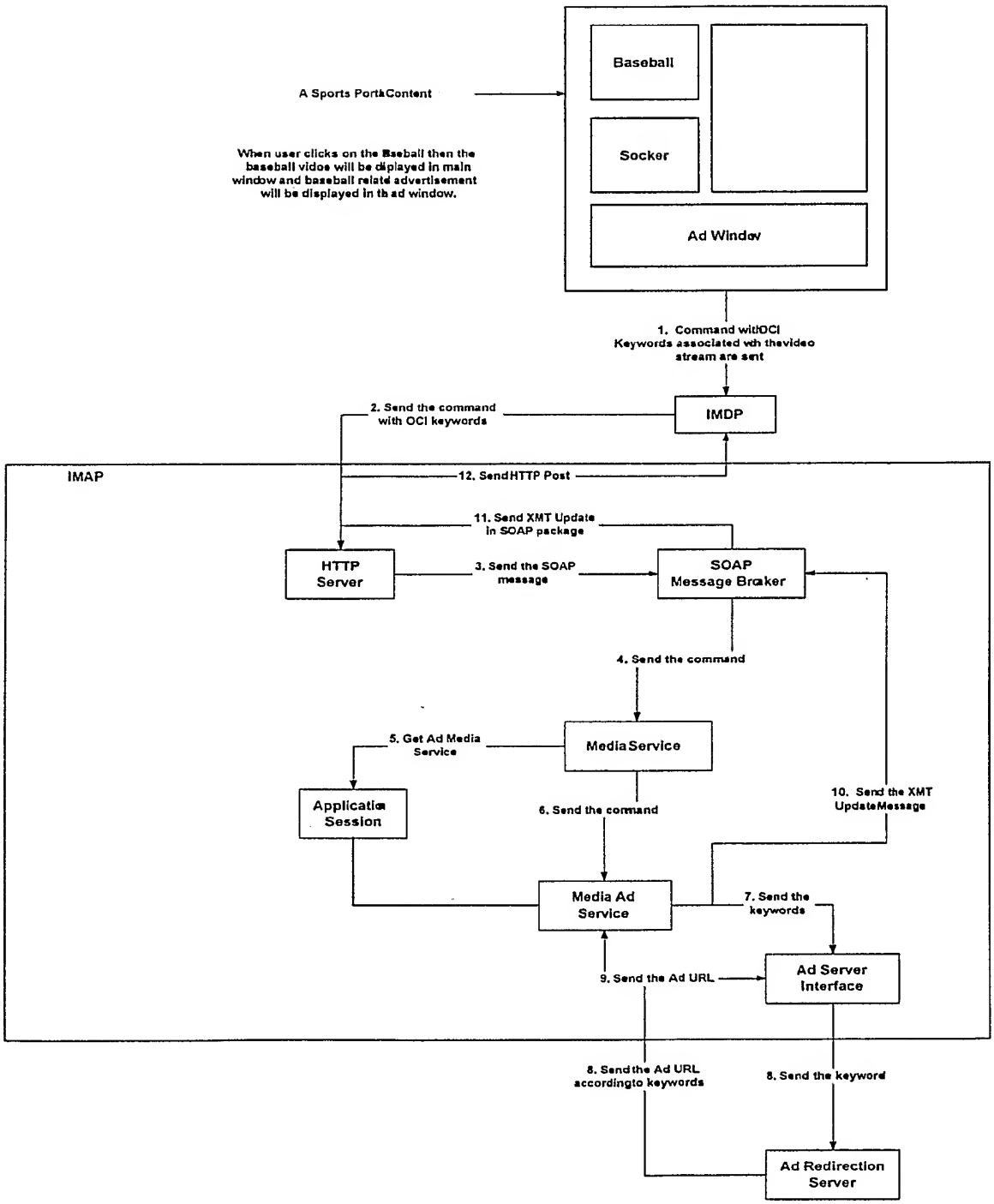
IMAP Provisional Patent Material



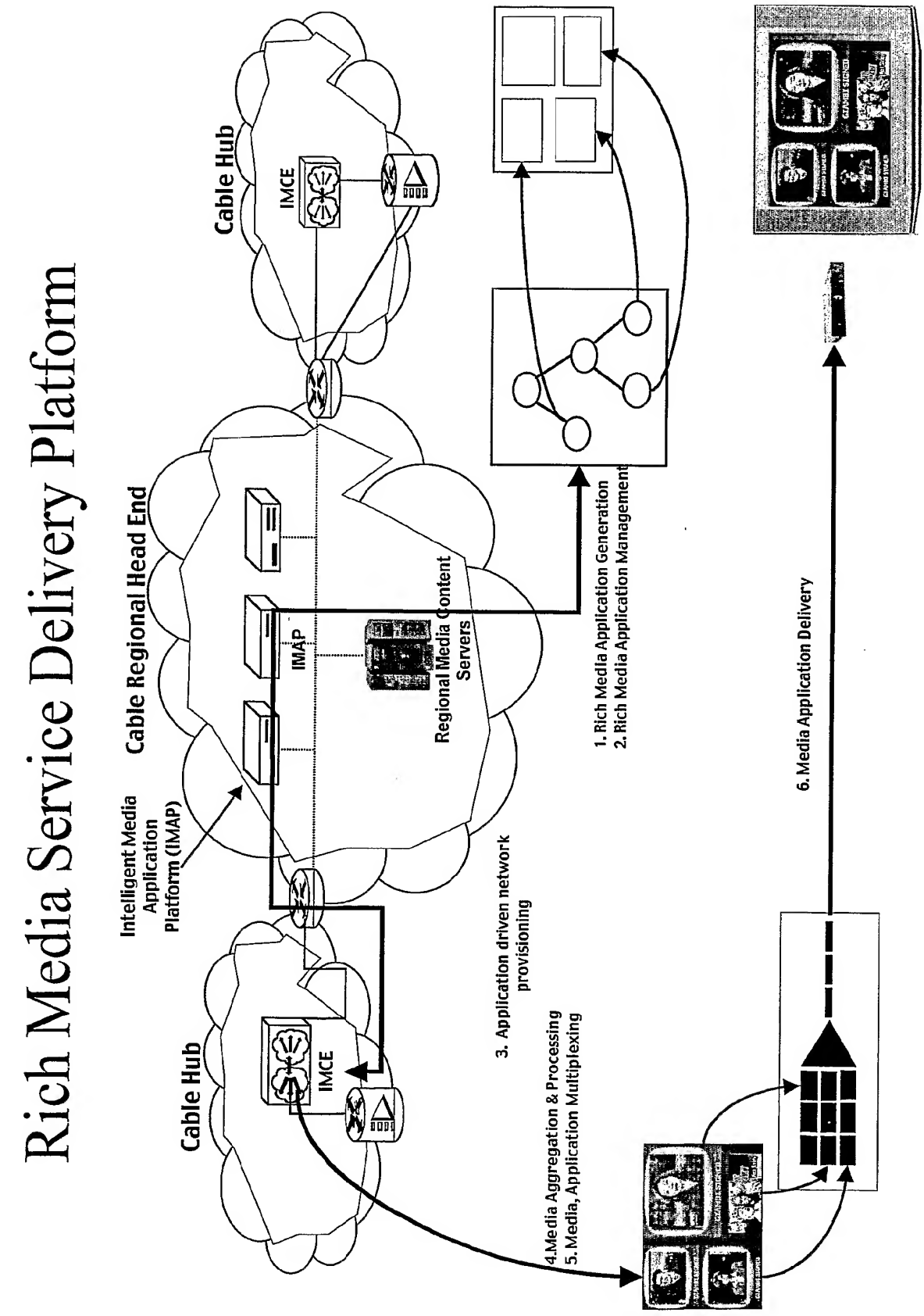

External Authoring Interface
1. Send a broadcast XMT message
Push Channel Interface
3. Get application session list
2. Get active service list
4. Send XMT message
Media Service Manager
Media Service
5. Create with unique graph node id
Application Session
7. Send SOAP Package
6. Package Scene Graph
XMT Scene Graph
SOAP Request Broaker
HTTP Server
8. Send SOAP package through HTTP post
IMAP
IMDP

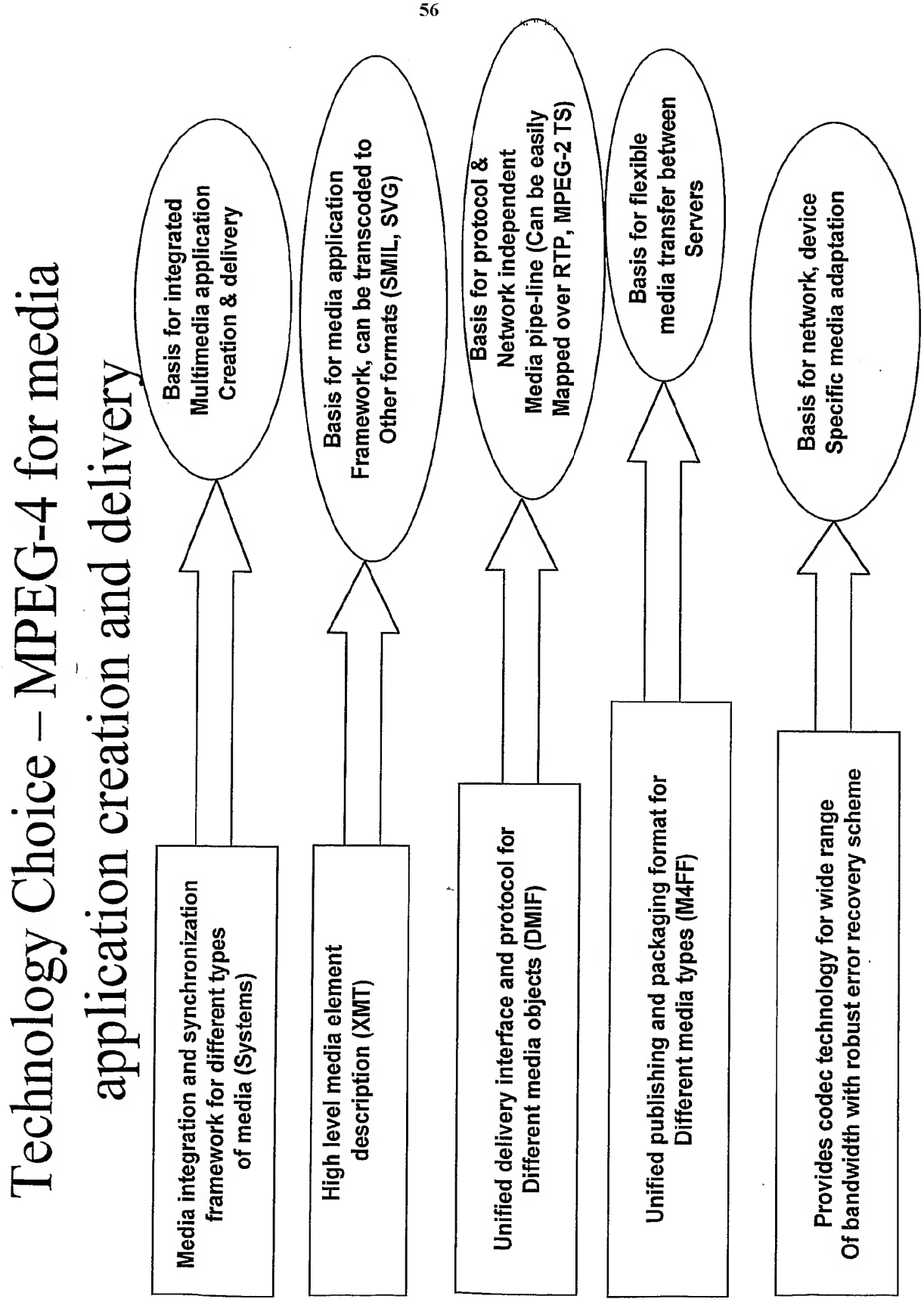

Ad Redirection Server
Media Service
Ad Service Interface
Application Session
2. Get cached User Profile
3. Get Ad Information (User Profile, Ad Server URL)
4. Send Ad redirection info (Stream URL, Duration)
XMT Banner Ad Node
HTTP Server
7. Send SOAP Package to IMDP
1. Ad duration timer expires
5. Reset Ad duration timer and Generate XMT Update Command
6. Package Update Command in SOAP
8. Send SOAP Packaged Update Message
SOAP Message Broaker
IMDP


text
OK
Buy
1. Send Buy Command (Using Server Command)
11. Encode XMT form in BIFS and Send via BIFS Channel
2. Send Buy Command (Element ID)
IMDP
IMAP
10. Send SOAP package in HTTP post
HTTP Server
3. Send Buy Command
SOAP Message Broaker
5. Handle Buy Command
4. Get Application Session (Session ID)
Media Service
8. Package XMT Form in SOAP (Update Command)
Application Session
6. Handle Buy Command
9. Send SOAP Package to IMDP
Commerce Service Element
7. Generate Credit Info Entry Form in XMT

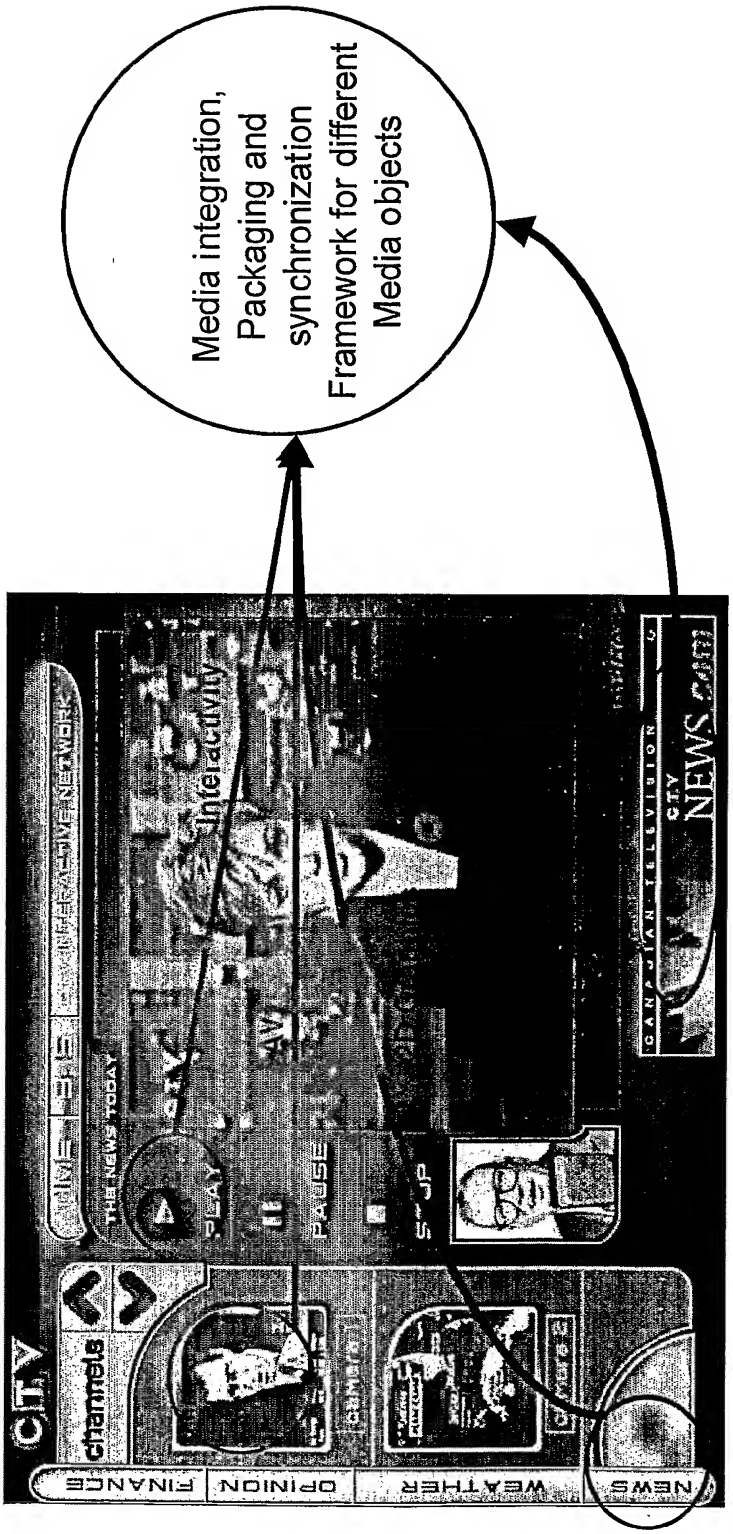

A Sports PortaContent
When user clicks on the Baseball then the baseball vidoe will be dipiayed in main window and baseball relate advertisement will be displayed in th ad window.
Baseball
Socker
Ad Window
1. Command witlOCI Keywords associated wh thevideo stream are sent
IMDP
2. Send the command with OCI keywords
IMAP
12. SendHTTP Post
11. Send XMT Update in SOAP package
HTTP Server
3. Send the SOAP message
SOAP Message Broker
4. Send the command
5. Get Ad Media Service
MediaService
Applicatio Session
6. Send the command
10. Send the XMT UpdateMessage
Media Ad Service
7. Send the keywords
9. Send the Ad URL
Ad Server Interface
8. Sendthe Ad URL accordingto keywords
8. Send the keyword
Ad Redirection Server

APPENDIX B

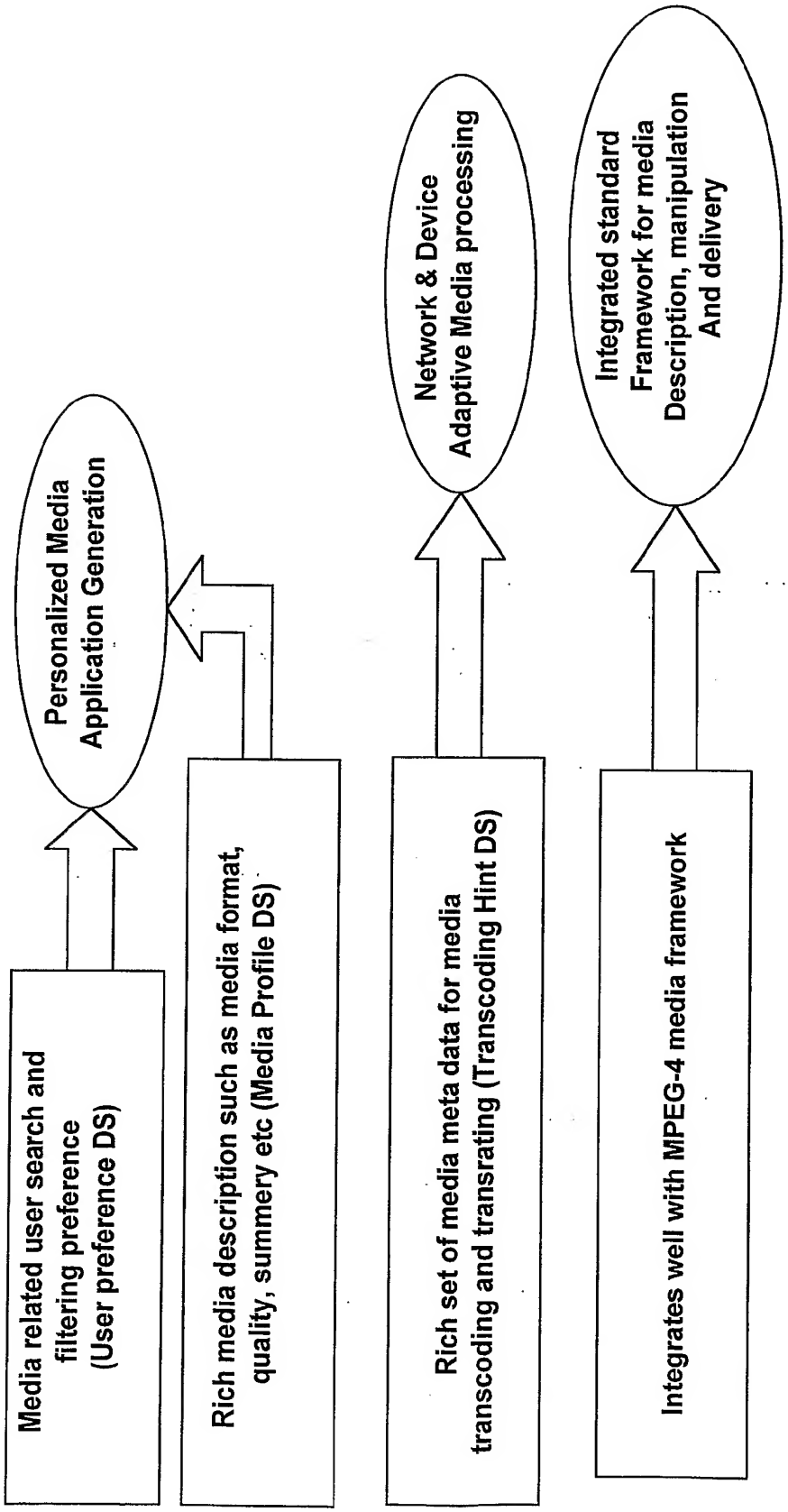

Rich Media Service Delivery Platform
Cable Hub
IMCE
Cable Regional Head End
IMAP
Regional Media Content Servers
Intelligent Media Application Platform (IMAP)
1. Rich Media Application Generation
2. Rich Media Application Management
3. Application driven network provisioning
4.Media Aggregation & Processing
5. Media, Application Multiplexing
6. Media Application Delivery

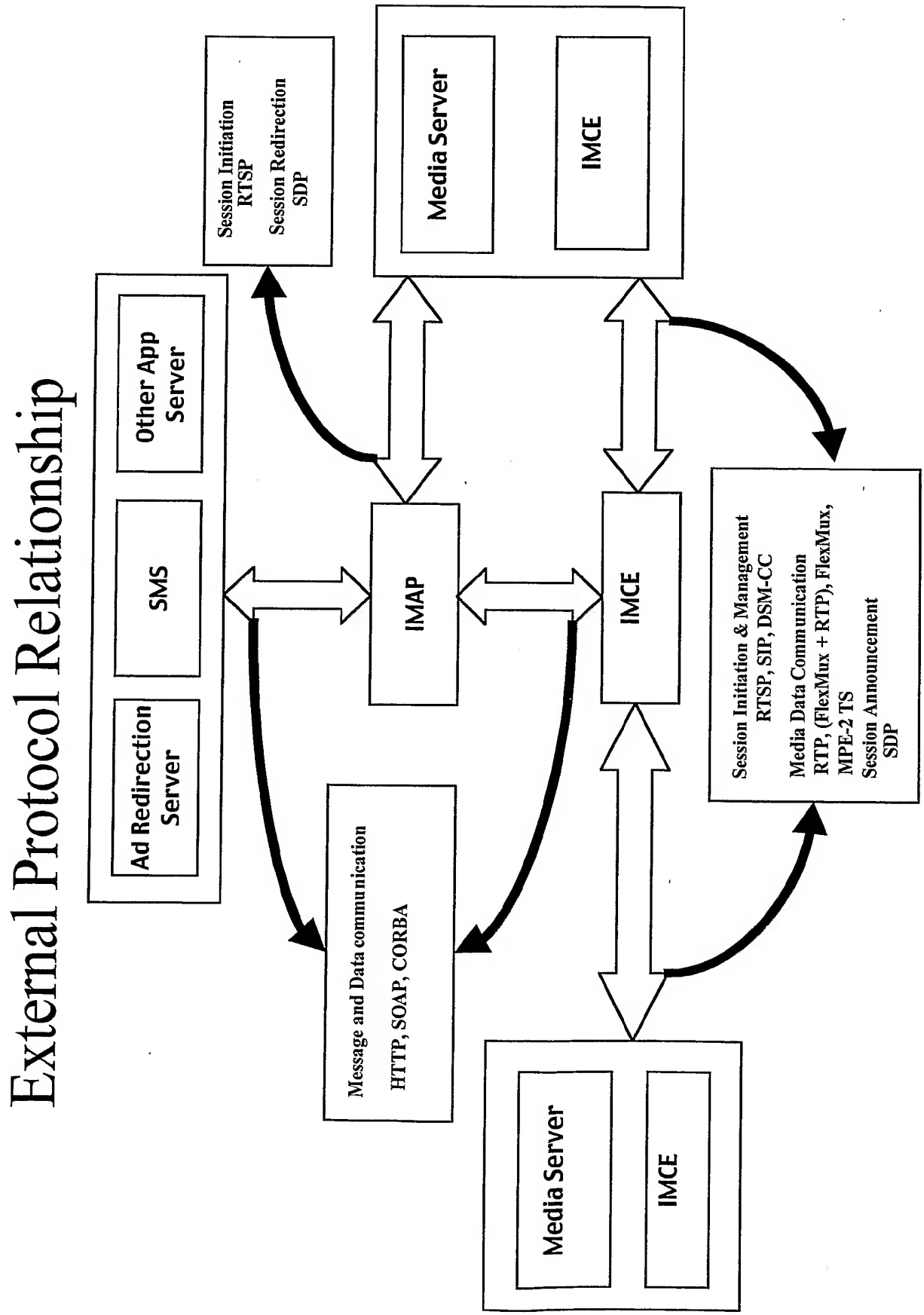

Technology Choice – MPEG-4 for media application creation and delivery
Media integration and synchronization framework for different types of media (Systems)
Basis for integrated Multimedia application Creation & delivery
High level media element description (XMT)
Basis for media application Framework, can be transcoded to Other formats (SMIL, SVG)
Unified delivery interface and protocol for Different media objects (DMIF)
Basis for protocol & Network independent Media pipe-line (Can be easily Mapped over RTP, MPEG-2 TS)
Unified publishing and packaging format for Different media types (M4FF)
Basis for flexible media transfer between Servers
Provides codec technology for wide range Of bandwidth with robust error recovery scheme
Basis for network, device Specific media adaptation

MPEG-4 Rich Media Service Creation and Delivery

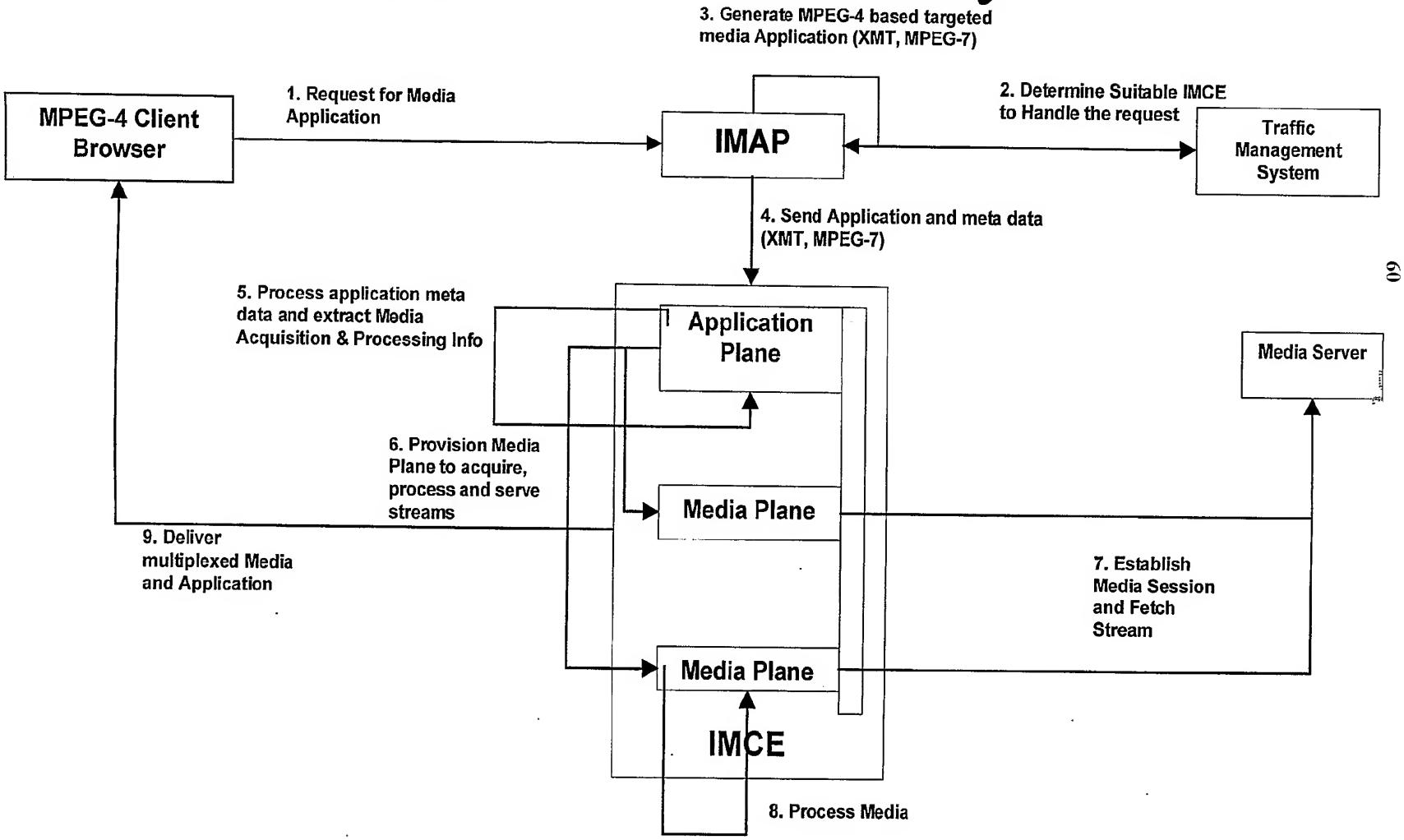


Technology Choice – MPEG-7 for media meta data description

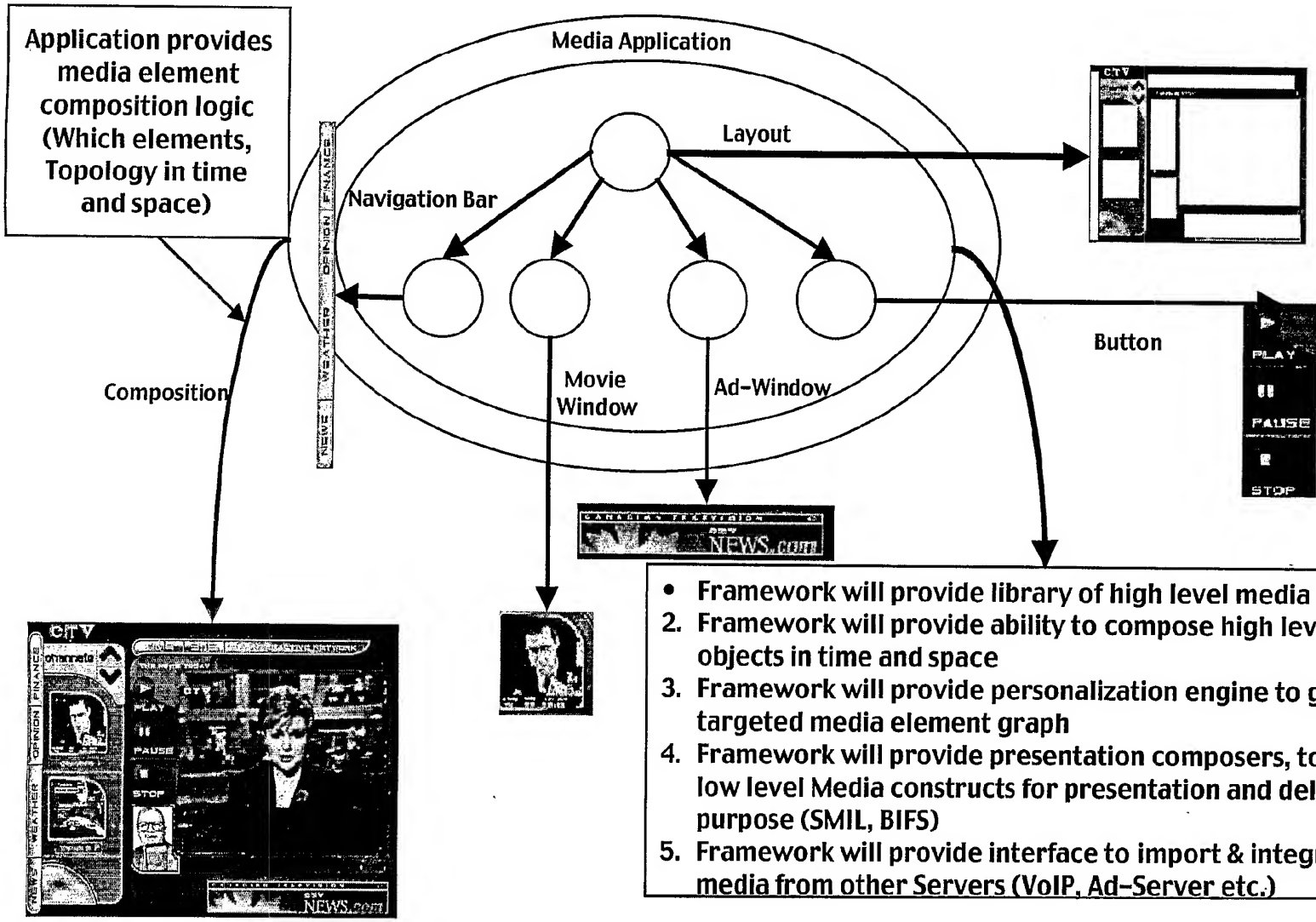



External Protocol Relationship
Session Initiation
RTSP
Session Redirection
SDP
Media Server
IMCE
Other App Server
SMS
Ad Redirection Server
IMAP
IMCE
Message and Data communication
HTTP, SOAP, CORBA
Session Initiation & Management
RTSP, SIP, DSM-CC
Media Data Communication
RTP, (FlexMux + RTP), FlexMux, MPE-2 TS
Session Announcement
SDP
Media Server
IMCE

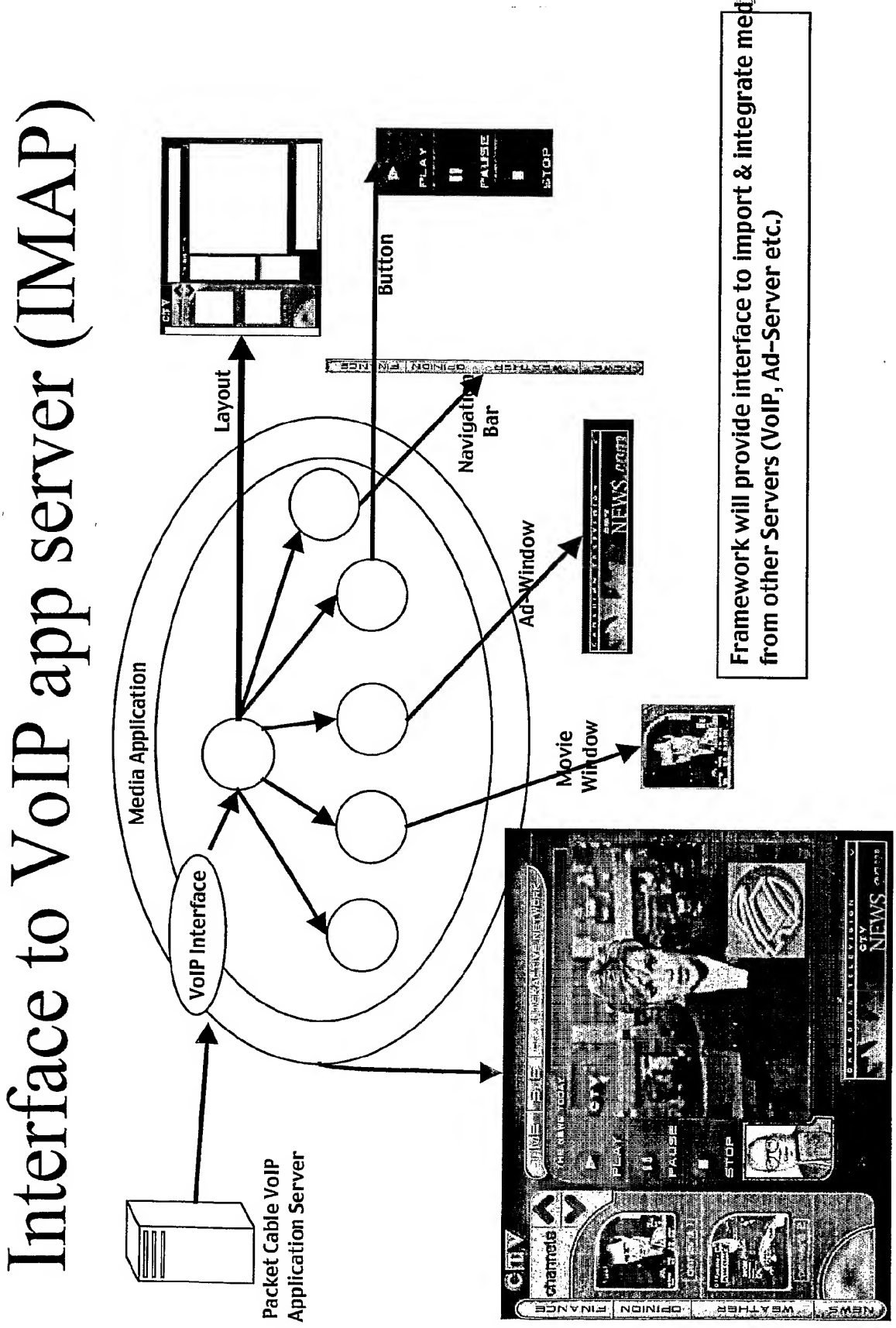
Data and control flow for media application delivery

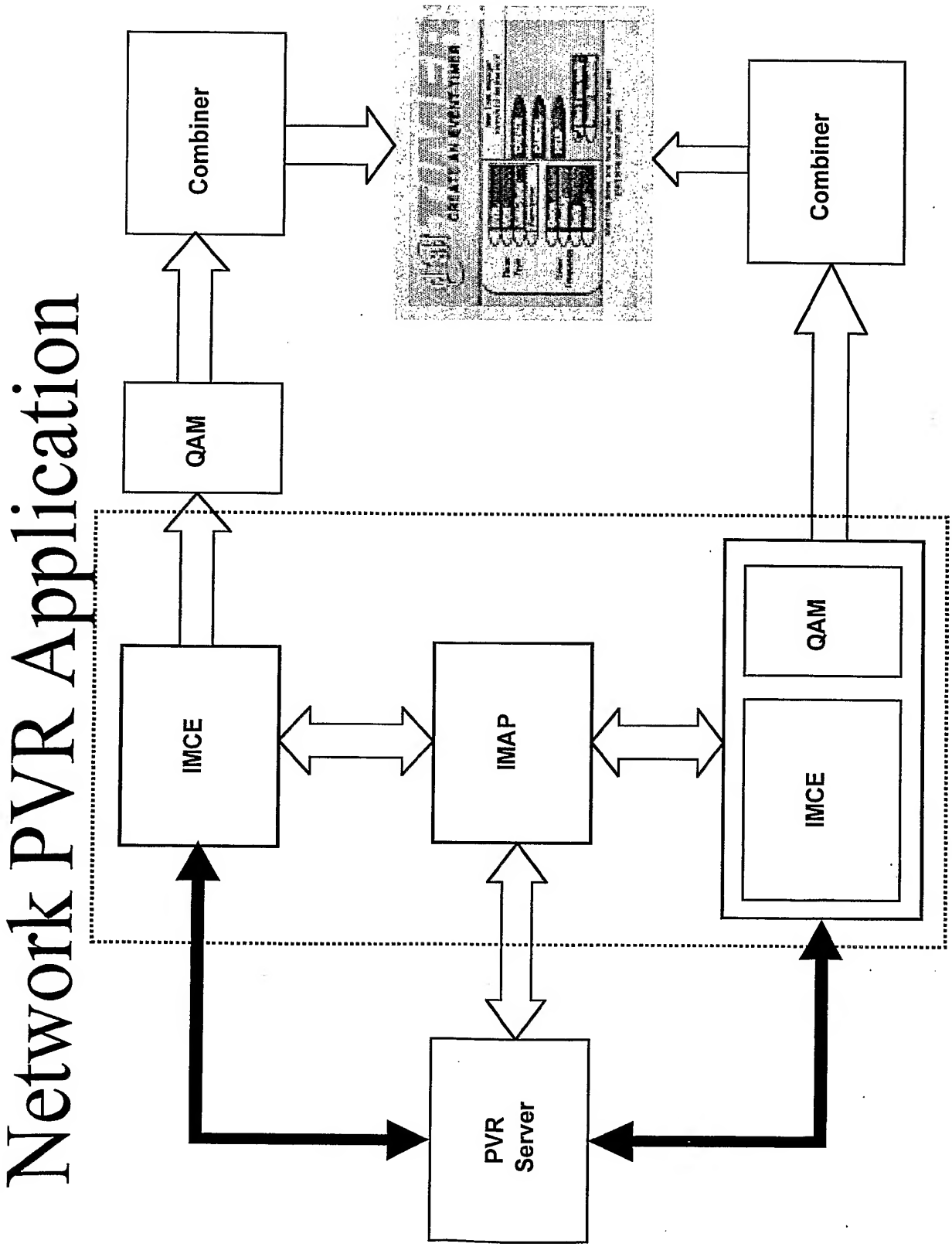


Media Application Framework (IMAP)

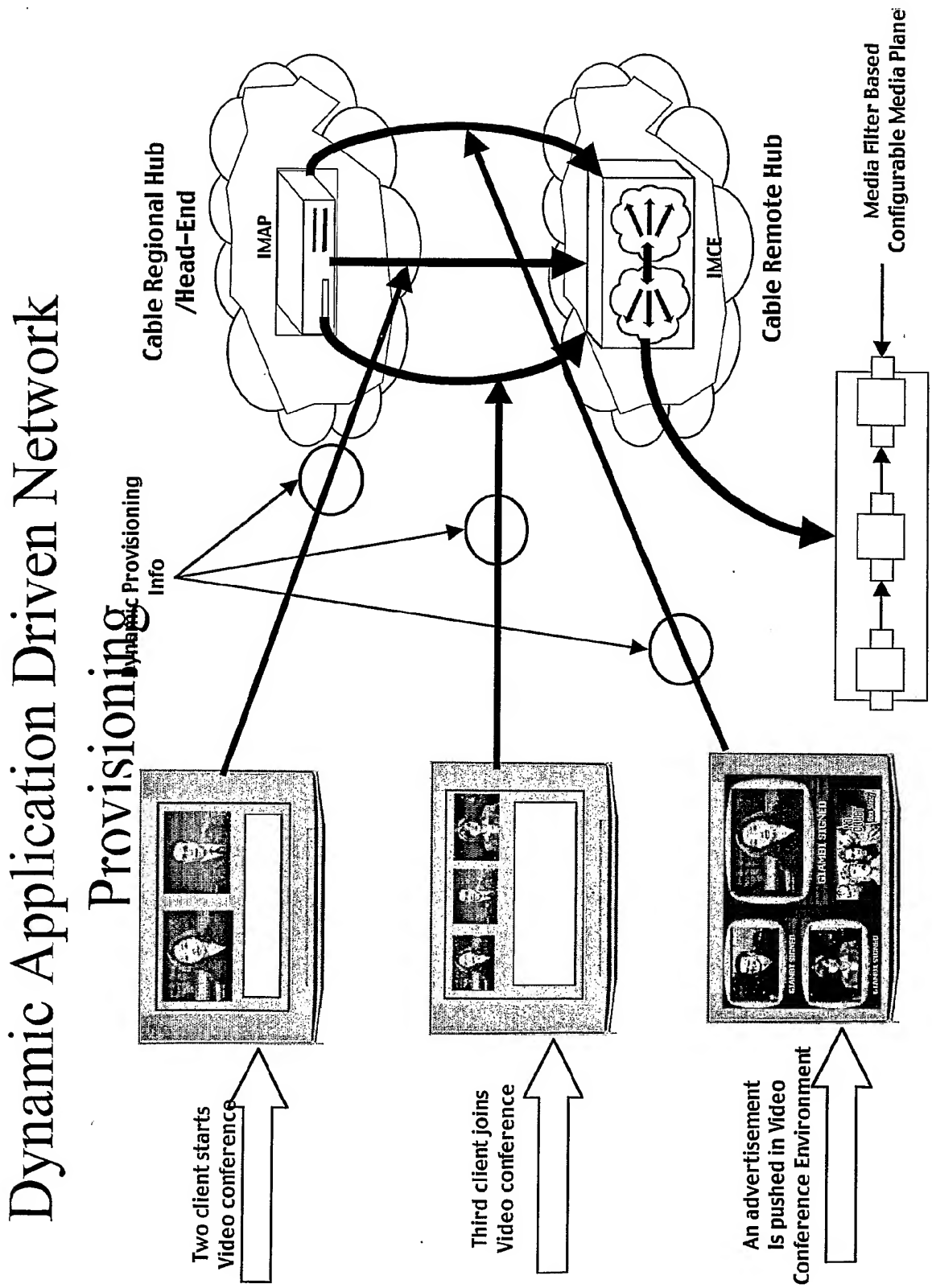


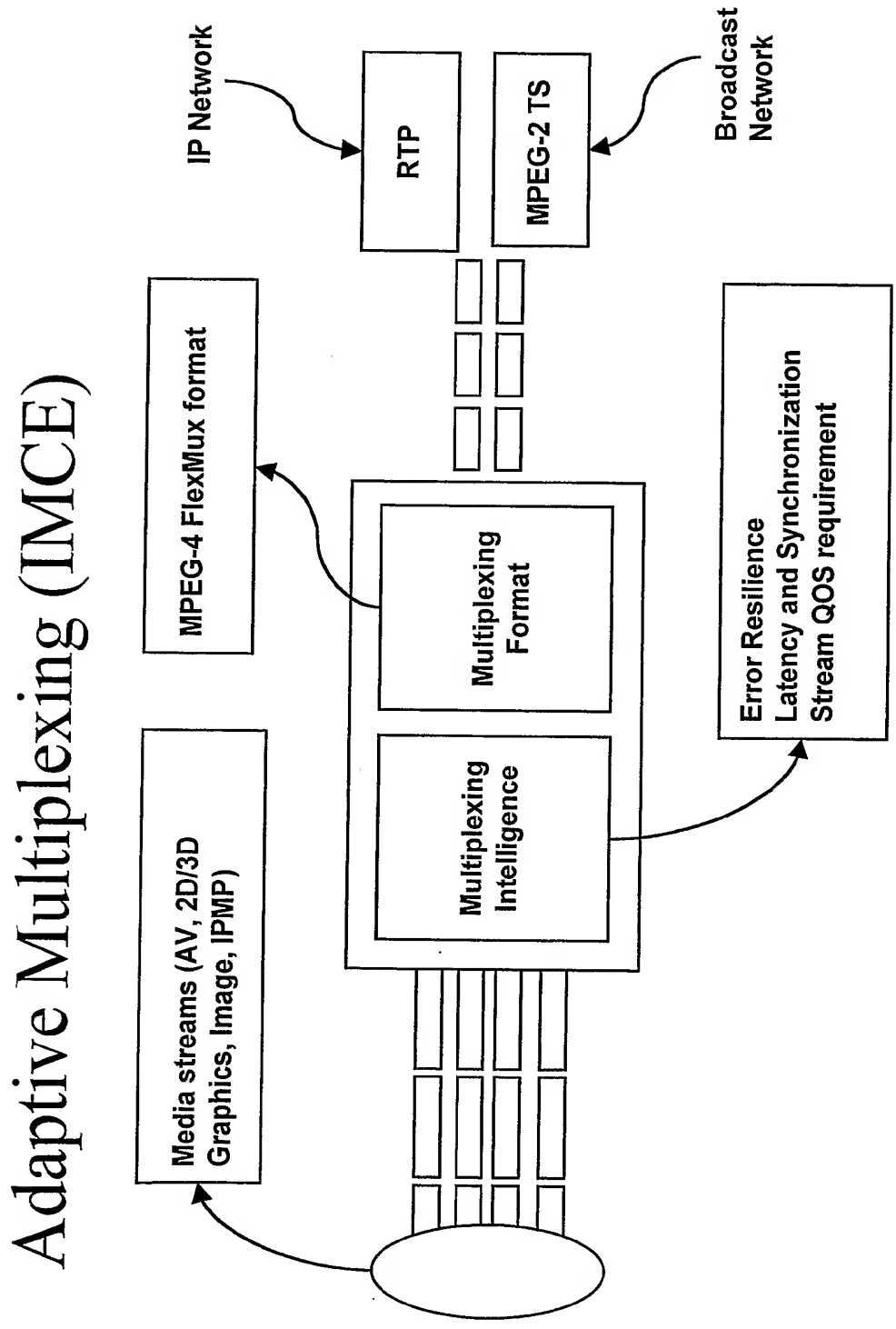
- Framework will provide library of high level media elements
2. Framework will provide ability to compose high level media objects in time and space
3. Framework will provide personalization engine to generate targeted media element graph
4. Framework will provide presentation composers, to generate low level Media constructs for presentation and delivery purpose (SMIL, BIFS)
5. Framework will provide interface to import & integrate media from other Servers (VoIP, Ad-Server etc.)


Interface to VoIP app server (IMAP)
Packet Cable VoIP Application Server
VoIP Interface
Media Application
Layout
Button
Navigation Bar
Ad-Window
Movie Window
Framework will provide interface to import & integrate media from other Servers (VoIP, Ad-Server etc.)

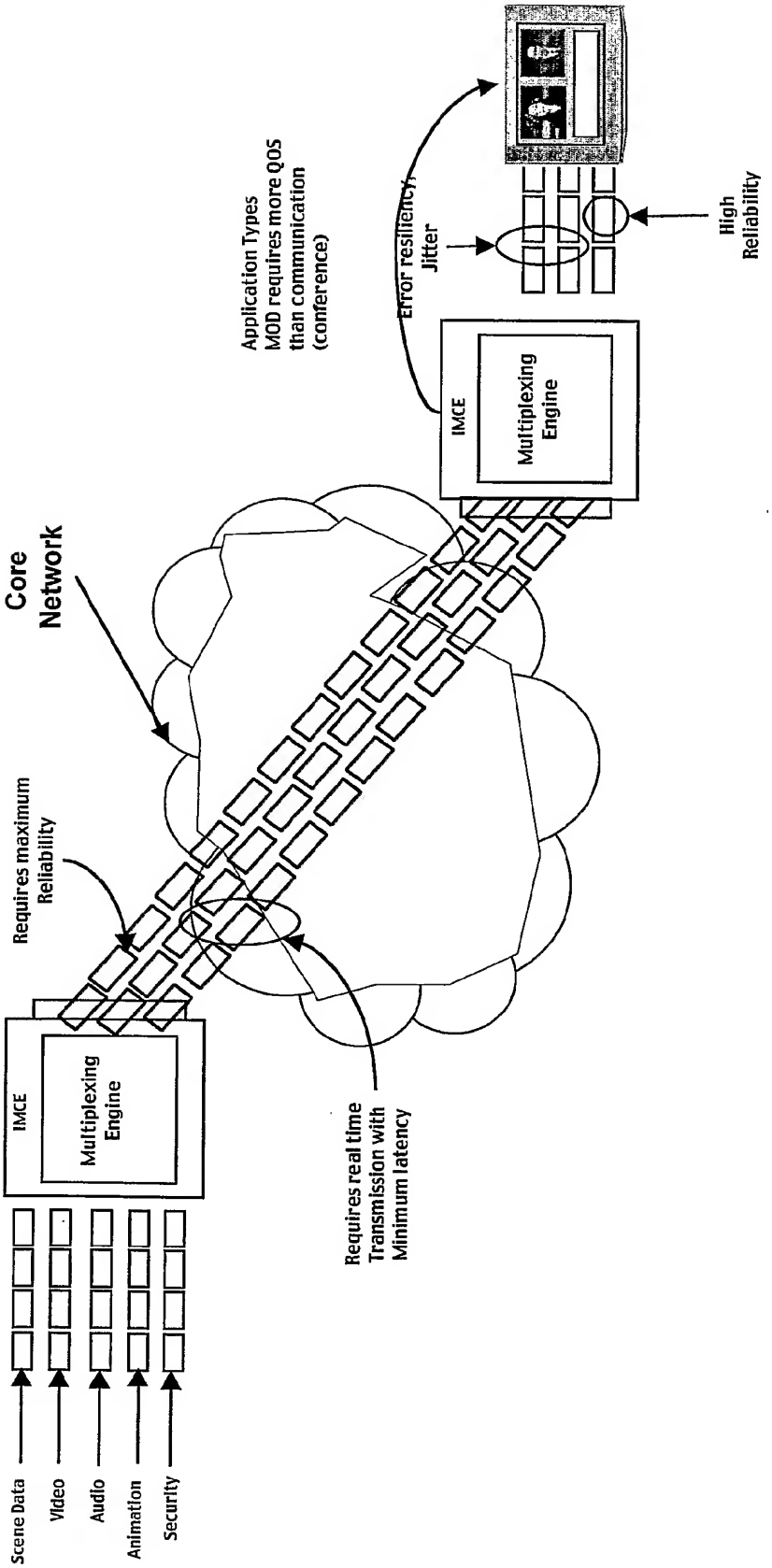

Network PVR Application
Combiner
Combiner
QAM
QAM
IMCE
IMAP
IMCE
PVR Server

Dynamic Media Application Generation (IMAP)
Rich Meta Data
<MP7: annotation>
<Who> Fernando </Who>
<What> Sports News </What>
<Source> cnn.mp4 </Source>
<MP7: annotation>
App Logic
<Java Program>
Object.func()
<XMT: Scene Element>
<video>
<video url = Cnn.mp4>
<transformation = "50 50">
</video>
<XMT: Scene Element>
User Profile
<MP7: annotation>
<User Preference>
<Preference> CNN Sports </Preference>
< Preference> Fox Finance </Preference
>
</User Preference>
<MP7: annotation>
Media Personalization Framework
Business Rule
<Rule>
If What = "Sports News" then
Then Ad keyword = "Sports"
</Rule>
Content Management System
User Management System
Rule Management System

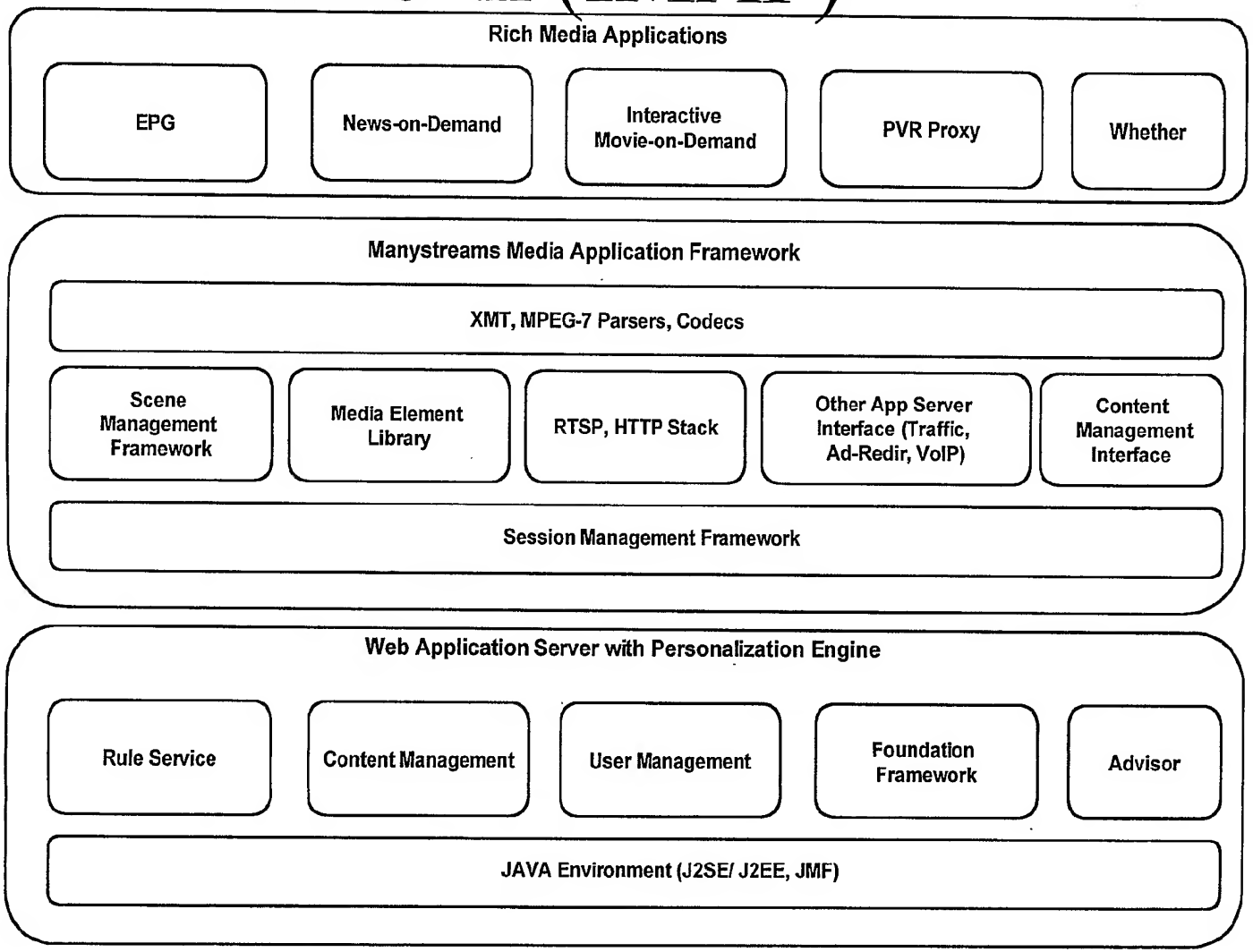

Dynamic Application Driven Network Provisioning
Dynamic Provisioning Info
Cable Regional Hub /Head-End
IMAP
IMCE
Cable Remote Hub
Media Filter Based Configurable Media Plane
Two client starts Video conference
Third client joins Video conference
An advertisement Is pushed in Video Conference Environment


Adaptive Multiplexing (IMCE)
IP Network
RTP
MPEG-2 TS
Broadcast Network
MPEG-4 FlexMux format
Multiplexing Format
Multiplexing Intelligence
Error Resilience
Latency and Synchronization
Stream QOS requirement
Media streams (AV, 2D/3D Graphics, Image, IPMP)

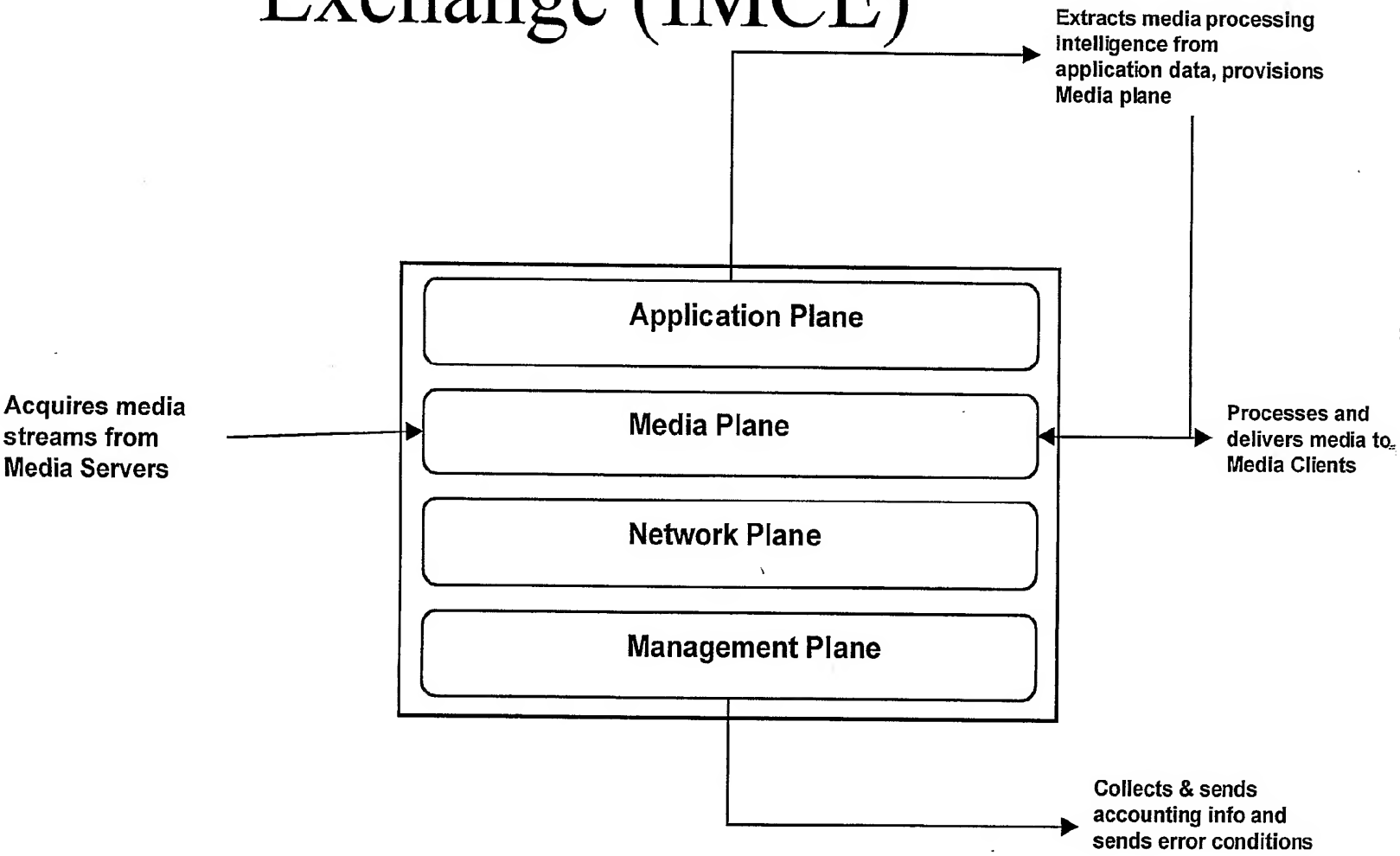
Adaptive Multiplexing (IMCE)



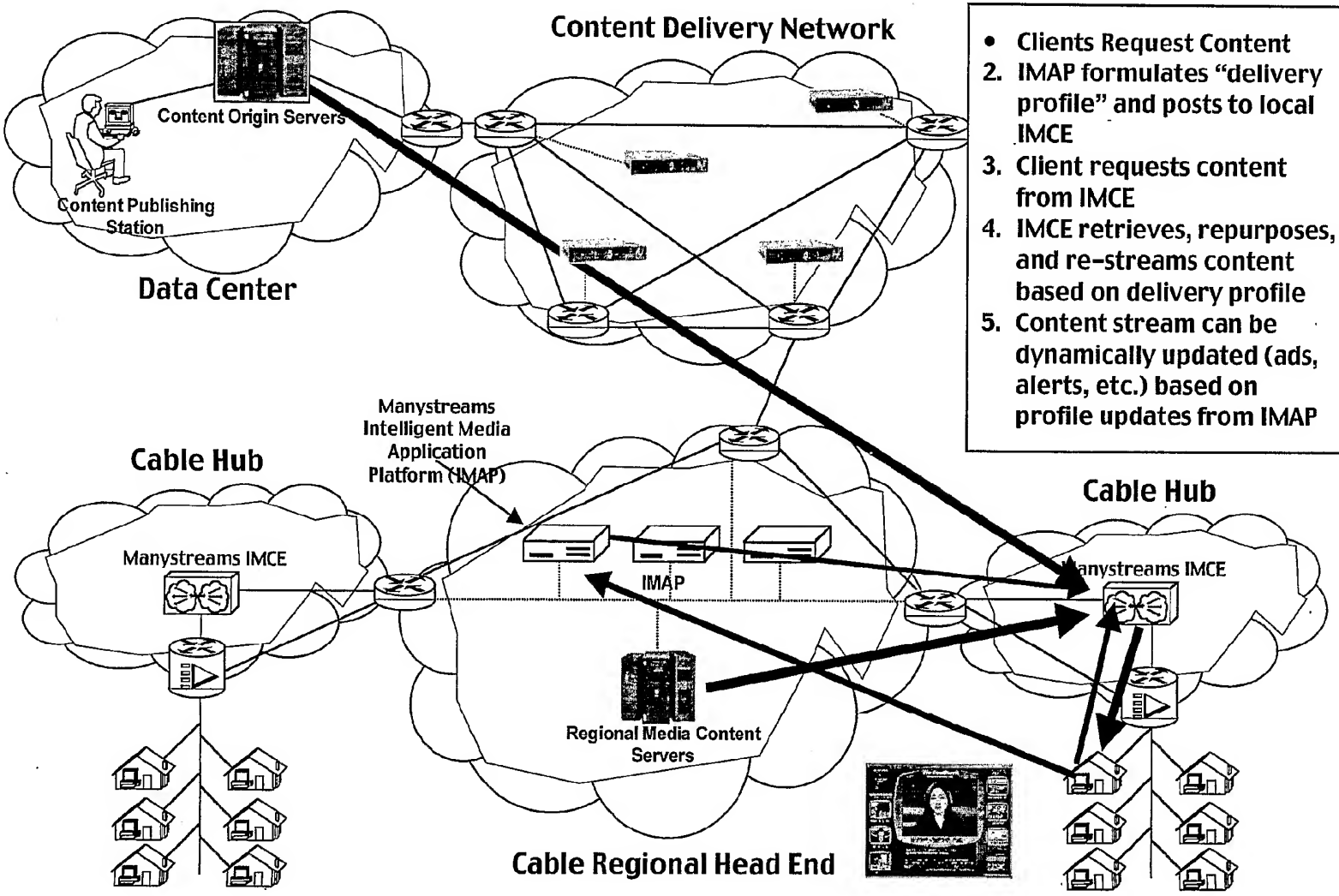
Intelligent Media Application Platform (IMAP)

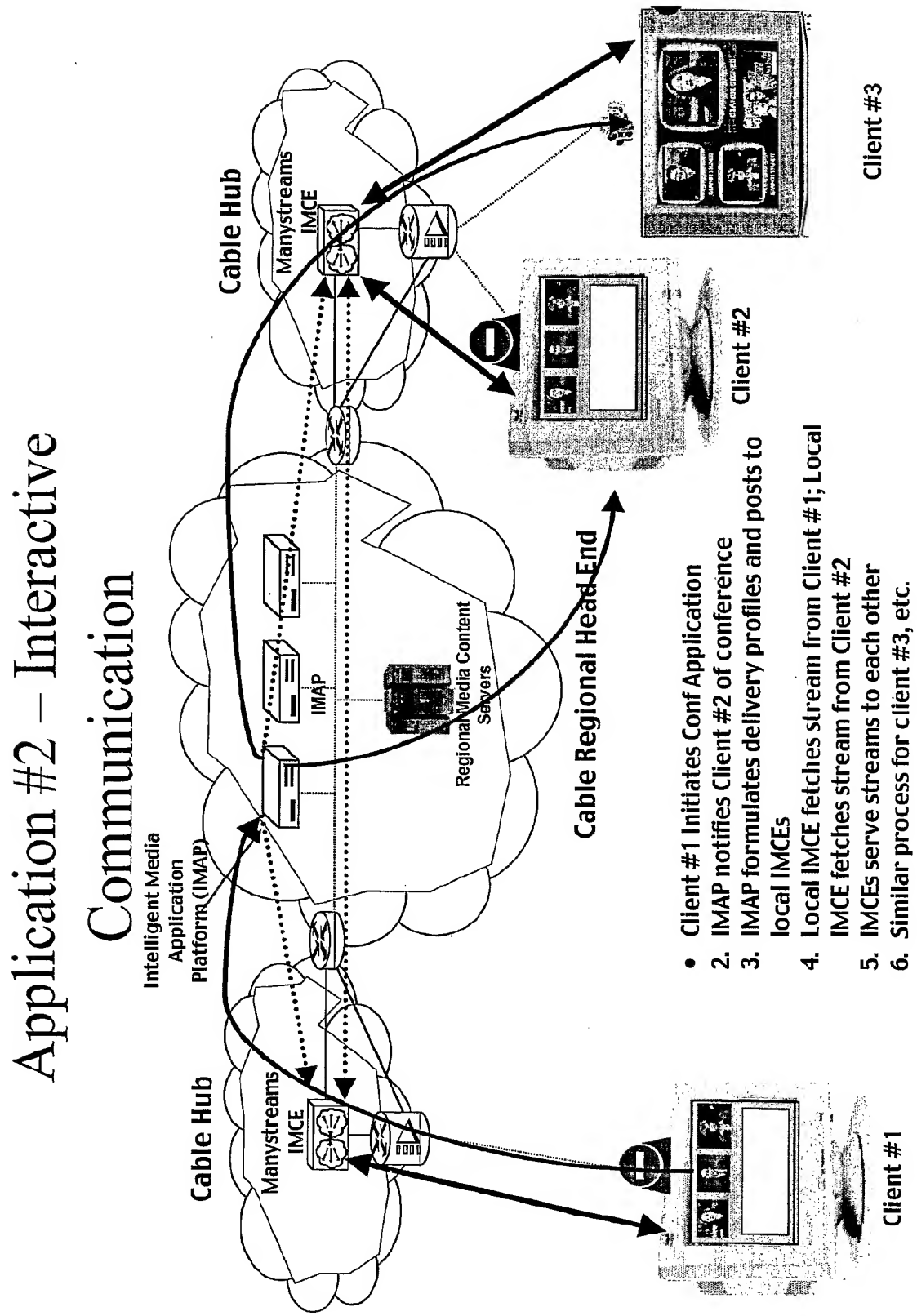


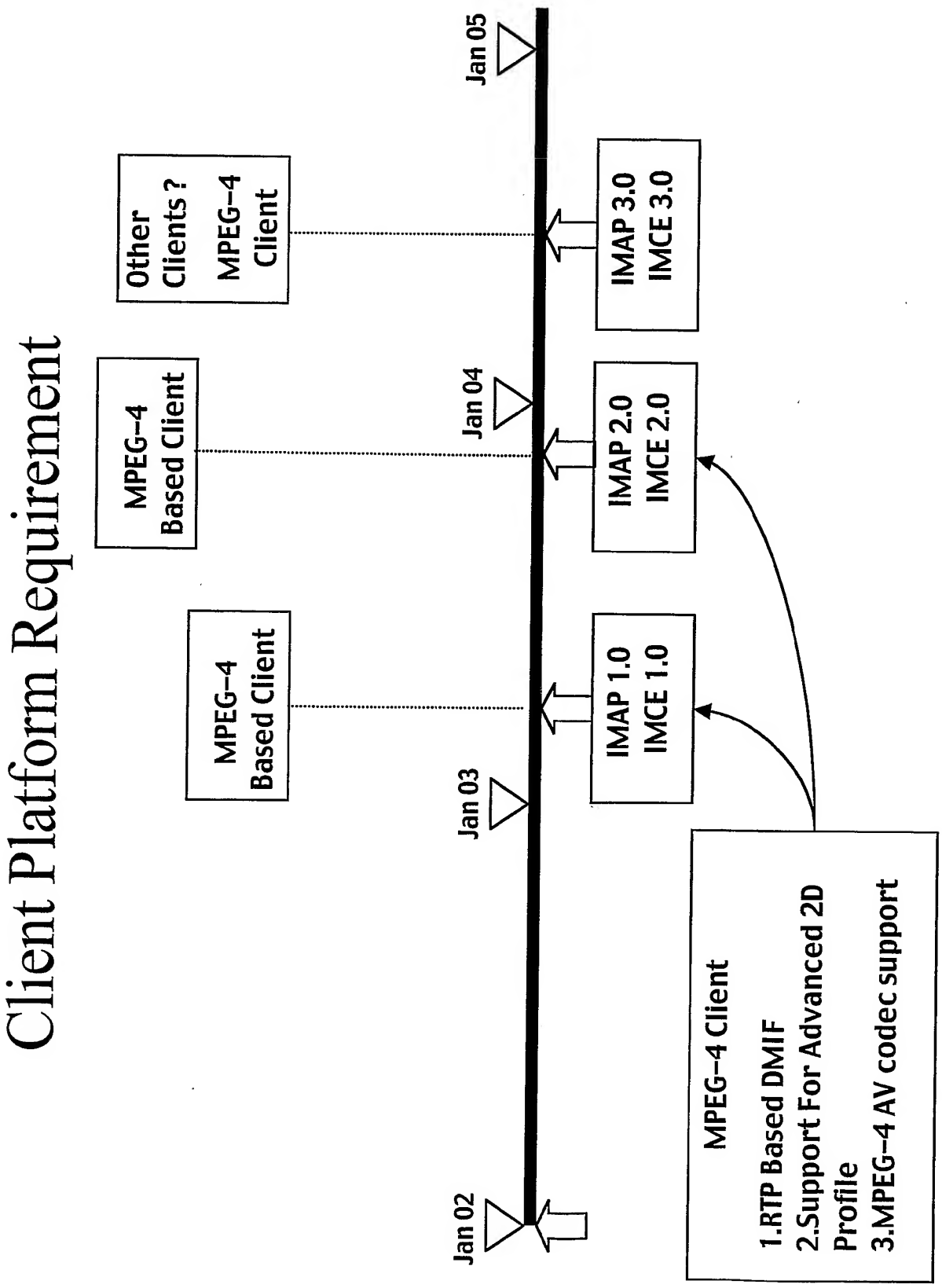
Intelligent Media Content Exchange (IMCE)



Application #1 – Store and Forward VOD Application




Application #2 – Interactive Communication
Intelligent Media Application Platform (IMAP)
Cable Hub
Manystreams IMCE
IMAP
Regional Media Content Servers
Cable Regional Head End
Cable Hub
Manystreams IMCE
Client #1
Client #2
Client #3
• Client #1 Initiates Conf Application
2. IMAP notifies Client #2 of conference
3. IMAP formulates delivery profiles and posts to local IMCEs
4. Local IMCE fetches stream from Client #1; Local IMCE fetches stream from Client #2
5. IMCEs serve streams to each other
6. Similar process for client #3, etc.


Client Platform Requirement
Jan 02
Jan 03
Jan 04
Jan 05
MPEG-4 Based Client
MPEG-4 Based Client
Other Clients ? MPEG-4 Client
IMAP 1.0 IMCE 1.0
IMAP 2.0 IMCE 2.0
IMAP 3.0 IMCE 3.0
MPEG-4 Client
1.RTP Based DMIF
2.Support For Advanced 2D Profile
3.MPEG-4 AV codec support

CLAIMS

What is claimed is:

1. An apparatus positioned at an edge of a network, comprising:

a bus;

a first line card coupled to the bus; and

a second line card coupled to the bus, the second line card adapted to handle acquisition of at least two different types of media content from different sources and to process the at least two different types of media content in order to integrate the at least two different types of media content into a single stream of media content.

2. The apparatus of claim 1 further comprising a third line card in communication with the second line card, the third line card being adapted for delivery of the single stream of media content to a remotely located client.

3. The apparatus of claim 2 being positioned at an edge of a content delivery network for transmission of the single stream of media content to the remotely located client.

4. The apparatus of claim 1, wherein the first line card is an application plane comprising a first parser to extract and separately route (1) information associated with presentation and (2) information associated with media processing.

5. The apparatus of claim 4, wherein the first parser of the application plane further extracting and separately routing service rights management data.

6. The apparatus of claim 5, wherein the first line card further comprises an interface and a plurality of parsers coupled to the first parser and the interface, the plurality of parsers generating commands for configuring functionality of the second line card.

7. The apparatus of claim 1 further comprising a back plane switch fabric coupled to the bus.

8. A method for integrating media content from a plurality of sources into a single media stream, the method comprising:

receiving incoming media content from the plurality of sources at an edge of a network;

processing the incoming media content into the single media steam at the edge of the network; and

delivering the media stream to a plurality of clients.

9. The method of claim 8, wherein the receiving of the media content comprises:

receiving a message with a data structure including information associated with presentation of the incoming media content and media processing hints; and

parsing the message to extract the information associated with the presentation of the incoming media content and the media processing hints to generate commands to establish a media processing pipeline of filters for processing the incoming media content.

10. The method of claim 9, wherein the media processing pipeline comprises a plurality of filters for processing the incoming media content and outputting outgoing media content, the plurality of filters includes a packet aggregator filter to aggregate incoming media content.

11. The method of claim 10, wherein the plurality of filters further comprises a transcoding filter to transcode the incoming media content of a first format into the outgoing media content having a second format differing from the first format.

12. The method of claim 11, wherein the first format is MPEG-2 and the second format is MPEG-4.

13. The method of claim 11, wherein the plurality of filters further comprises a transrating filter to adjust a transfer frame rate from a difference between the incoming media content and the outgoing media content.

14. The method of claim 11, wherein the plurality of filters further comprises a decryption filter to decrypt the incoming media content.

15. The method of claim 14, wherein the plurality of filters further comprises an encryption filter to encrypt the outgoing media content.

16. Stored in a machine readable medium and executed by a processor positioned at an edge of a network, application driven software comprising:

a first module to handle acquisition of at least two different types of media content from different sources; and

a second module to process the at least two different types of media content in order to integrate the at least two different types of media content into a single stream of media content.

17. The application driven software of claim 16 further comprising a third module to deliver the single stream of media content to a remotely located client.

18. The application driven software of claim 17 further comprising a media manager to interpret incoming information received by an application server and to configure the first, second and third modules via a Common Object Request Broker Architecture (CORBA) API.

19. The application driven software of claim 17, wherein the first, second and third modules exchange control information using Common Object Request Broker Architecture (CORBA) messages. The modules 321-323 use inter-process communication (IPC) mechanisms such as sockets to exchange media content.

20. The application driven software of claim 17, wherein the first, second and third modules exchange media content using inter-process communication (IPC) mechanisms inclusive of sockets.

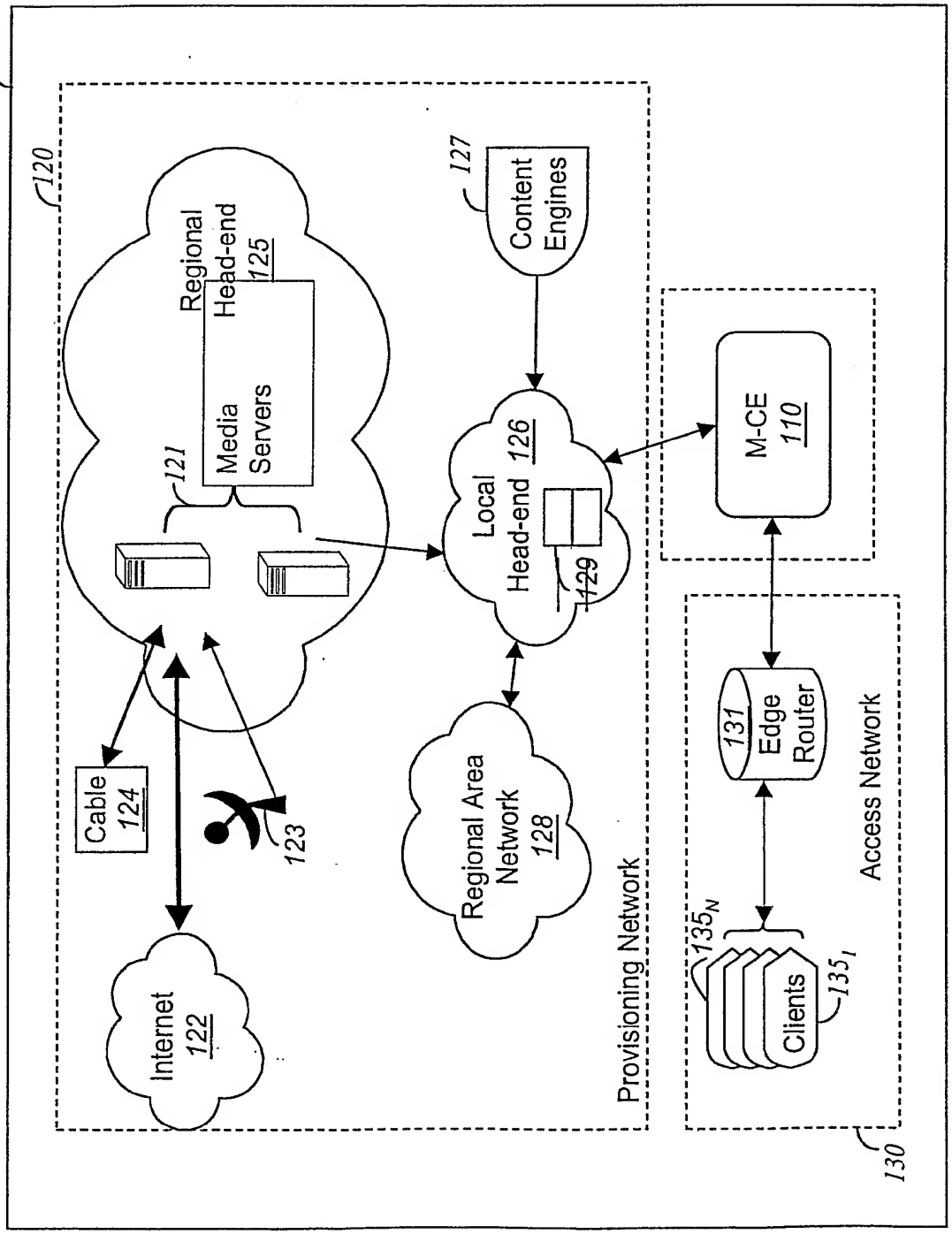

100
120
Regional Head-end 125
Media Servers
121
Content Engines
127
Local Head-end 126
129
M-CE 110
Cable 124
123
Regional Area Network 128
Internet 122
Provisioning Network
131
Edge Router
135_N
135_1
Clients
Access Network
130

Figure 1

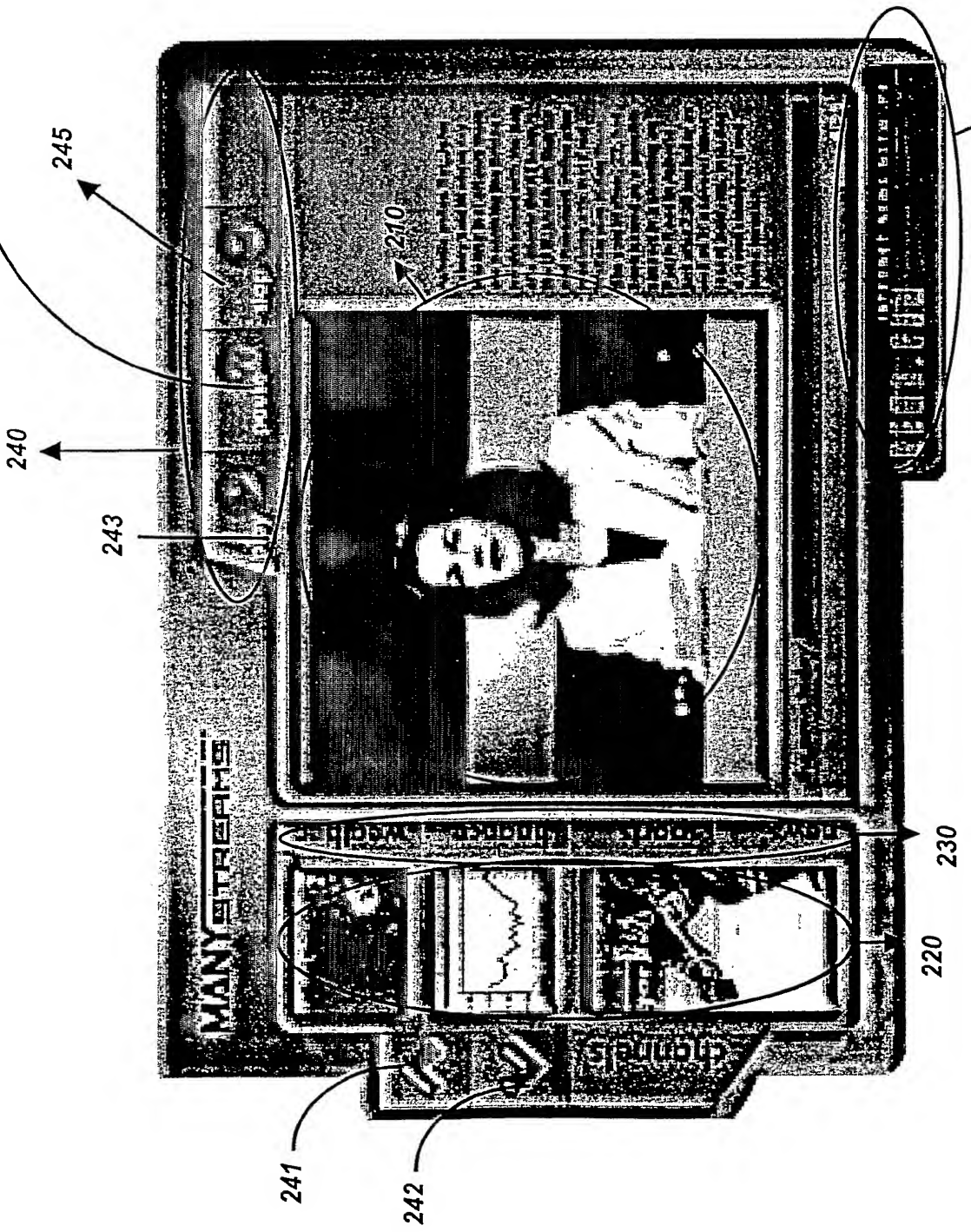


Figure 2

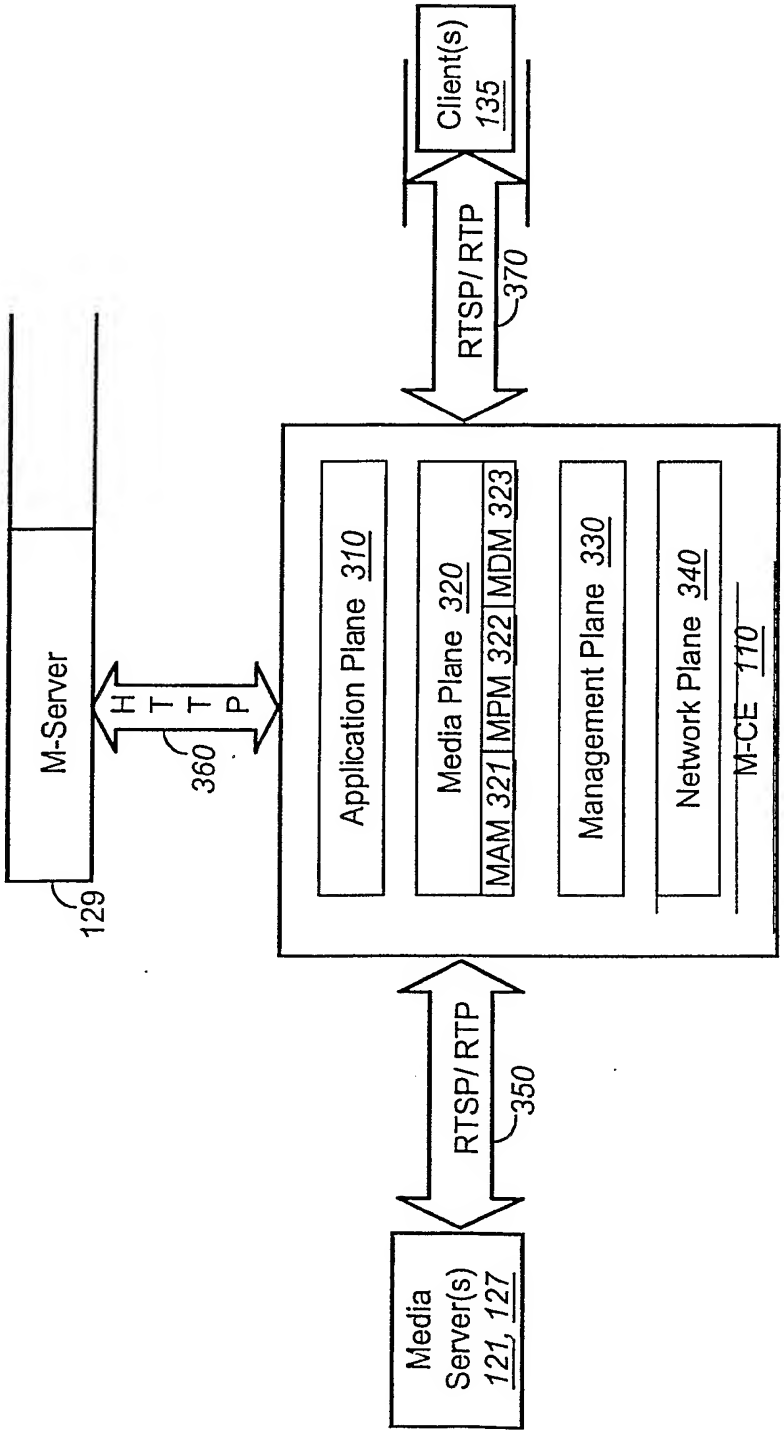

Client(s) 135
RTSP/ RTP
370
M-Server
129
H T T P
360
Application Plane 310
Media Plane 320
MAM 321
MPM 322
MDM 323
Management Plane 330
Network Plane 340
M-CE 110
RTSP/ RTP
350
Media Server(s) 121, 127

Figure 3

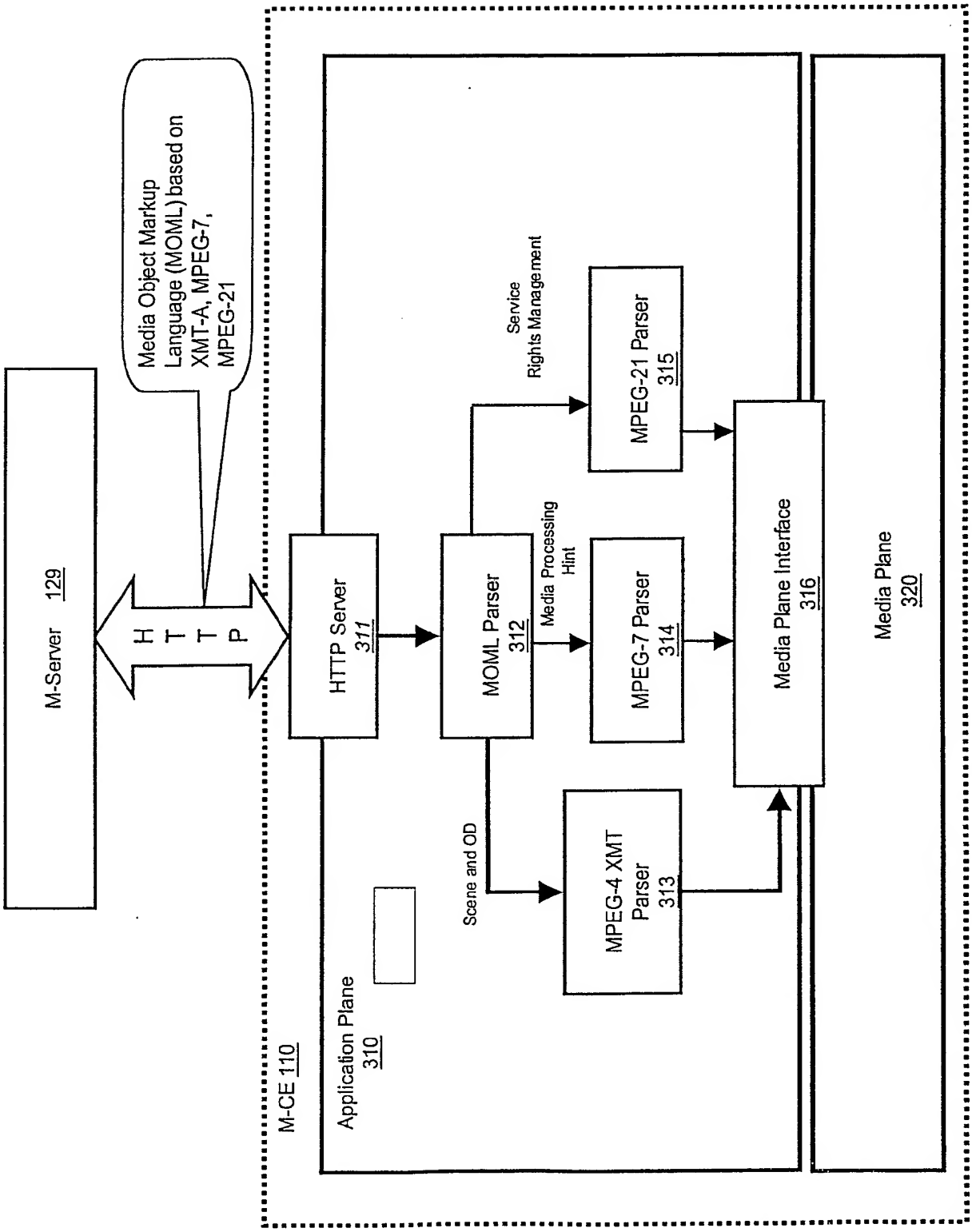

M-Server 129
H T T P
Media Object Markup Language (MOML) based on XMT-A, MPEG-7, MPEG-21
M-CE 110
Application Plane 310
HTTP Server 311
MOML Parser 312
Media Processing Hint
Service Rights Management
MPEG-21 Parser 315
MPEG-7 Parser 314
MPEG-4 XMT Parser 313
Scene and OD
Media Plane Interface 316
Media Plane 320

Figure 4

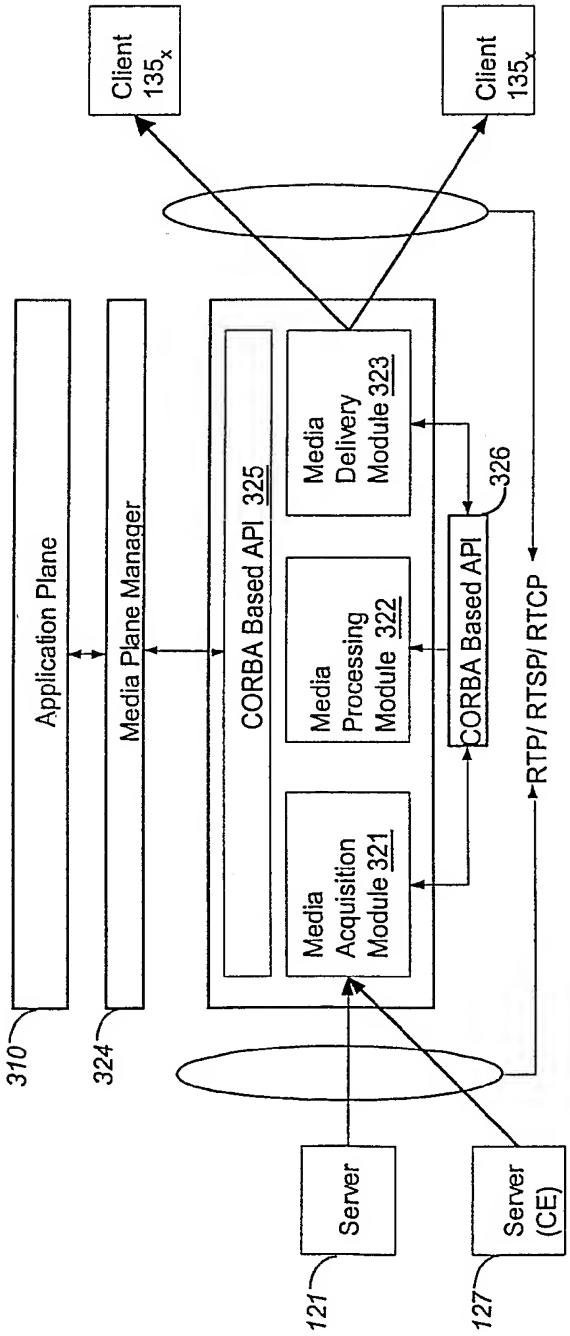

Client 135x
Client 135x
Application Plane
Media Plane Manager
CORBA Based API 325
Media Acquisition Module 321
Media Processing Module 322
Media Delivery Module 323
CORBA Based API
326
RTP/ RTSP/ RTCP
310
324
Server
Server (CE)
121
127

Figure 5

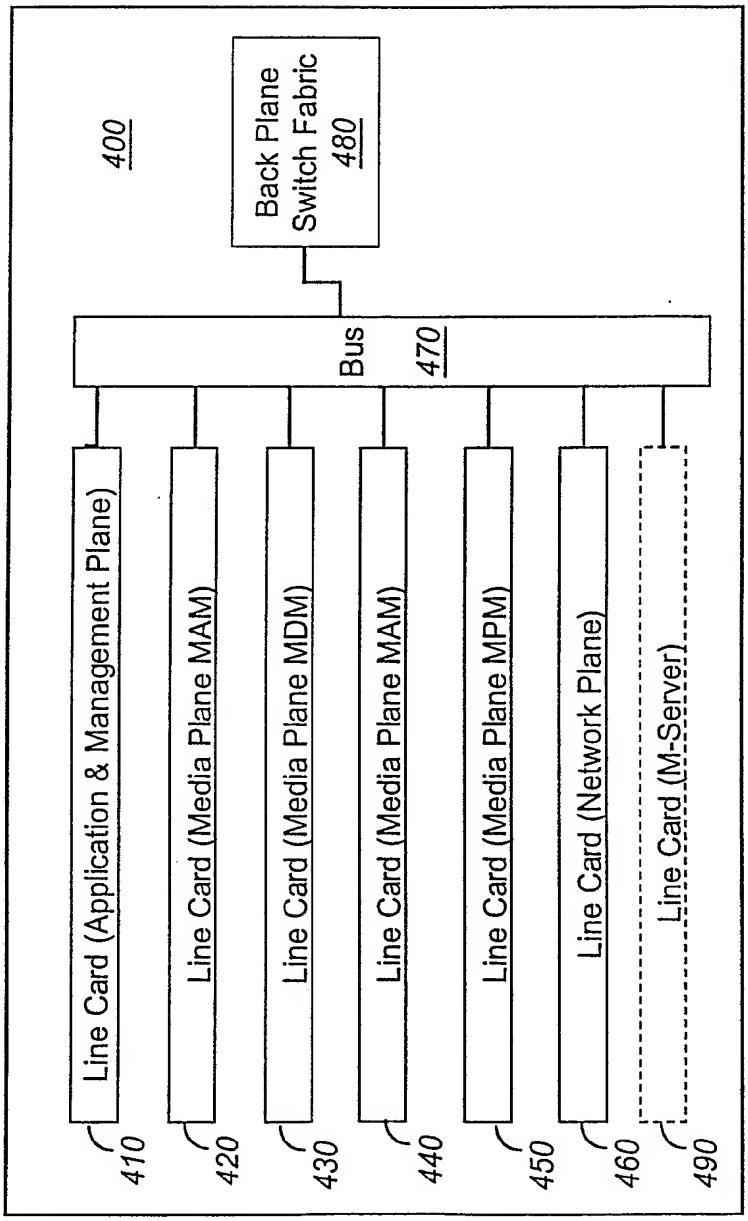

400
Back Plane Switch Fabric 480
Bus 470
Line Card (Application & Management Plane) 410
Line Card (Media Plane MAM) 420
Line Card (Media Plane MDM) 430
Line Card (Media Plane MAM) 440
Line Card (Media Plane MPM) 450
Line Card (Network Plane) 460
Line Card (M-Server) 490

Figure 6

Figure 7

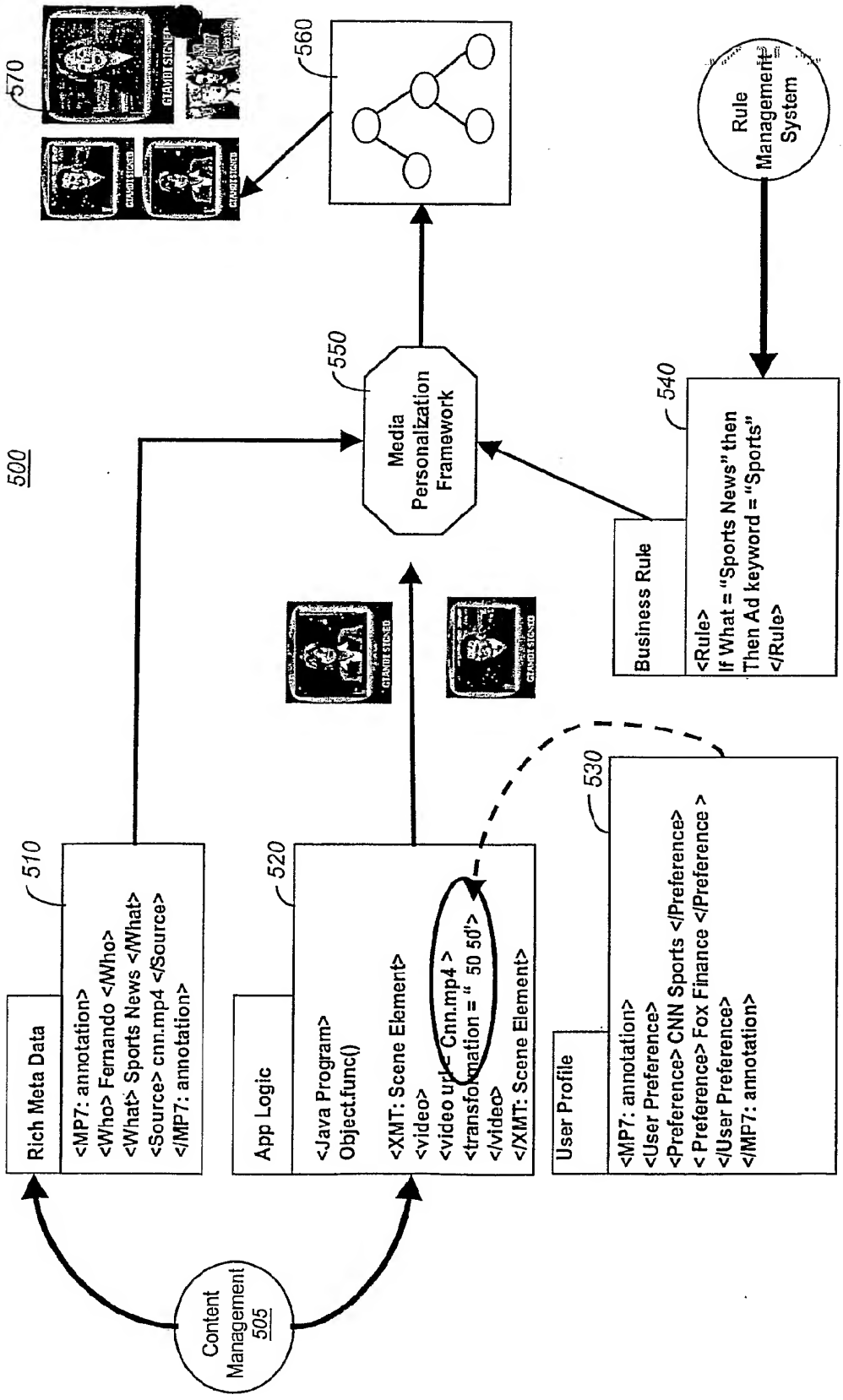
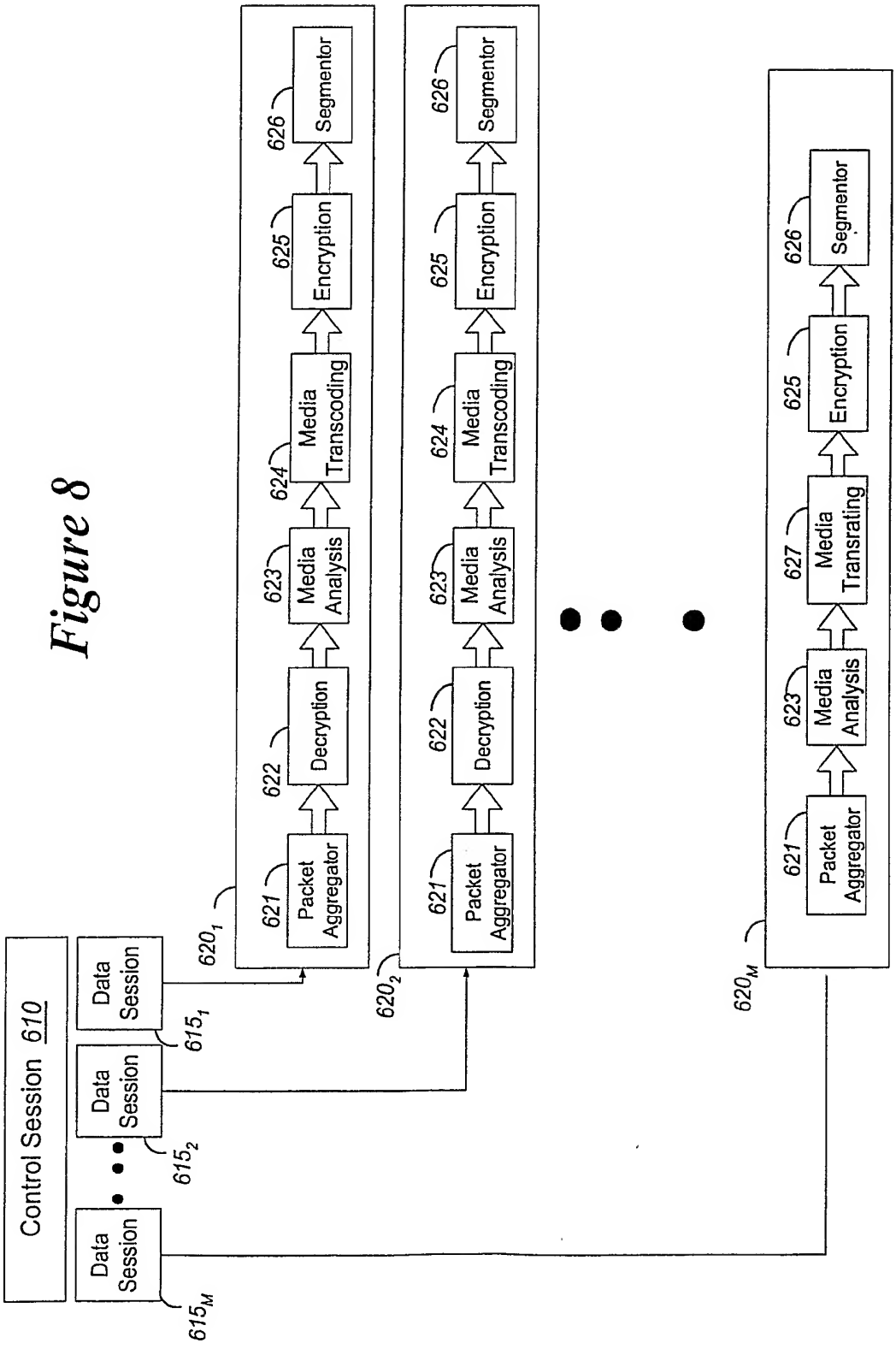

500
570
560
550
Media Personalization Framework
540
Business Rule
<Rule>
If What = "Sports News" then
Then Ad keyword = "Sports"
</Rule>
Rule Management System
510
Rich Meta Data
<MP7: annotation>
<Who> Fernando </Who>
<What> Sports News </What>
<Source> cnn.mp4 </Source>
</MP7: annotation>
520
App Logic
<Java Program>
Object.func()
<XMT: Scene Element>
<video>
<video url = Cnn.mp4 >
<transformation = " 50 50">
</video>
</XMT: Scene Element>
530
User Profile
<MP7: annotation>
<User Preference>
<Preference> CNN Sports </Preference>
< Preference> Fox Finance </Preference >
</User Preference>
</MP7: annotation>
Content Management
505

Figure 8


Control Session 610
Data Session
615M
Data Session
615₂
Data Session
615₁
620₁
620₂
620M
621 Packet Aggregator
622 Decryption
623 Media Analysis
624 Media Transcoding
625 Encryption
626 Segmentor
627 Media Transrating

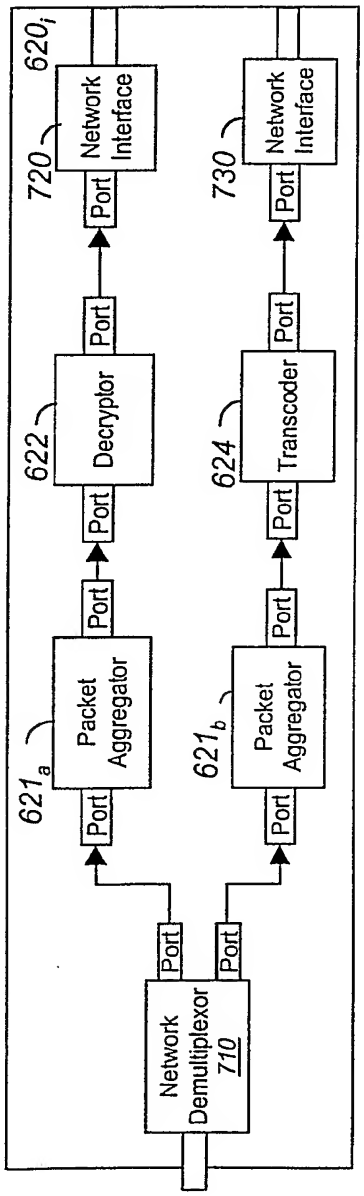


Figure 9

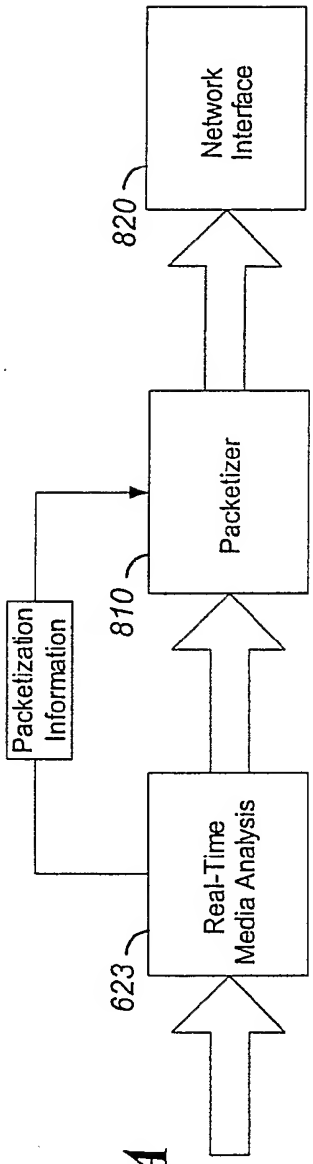


Figure 10A

Figure 10B

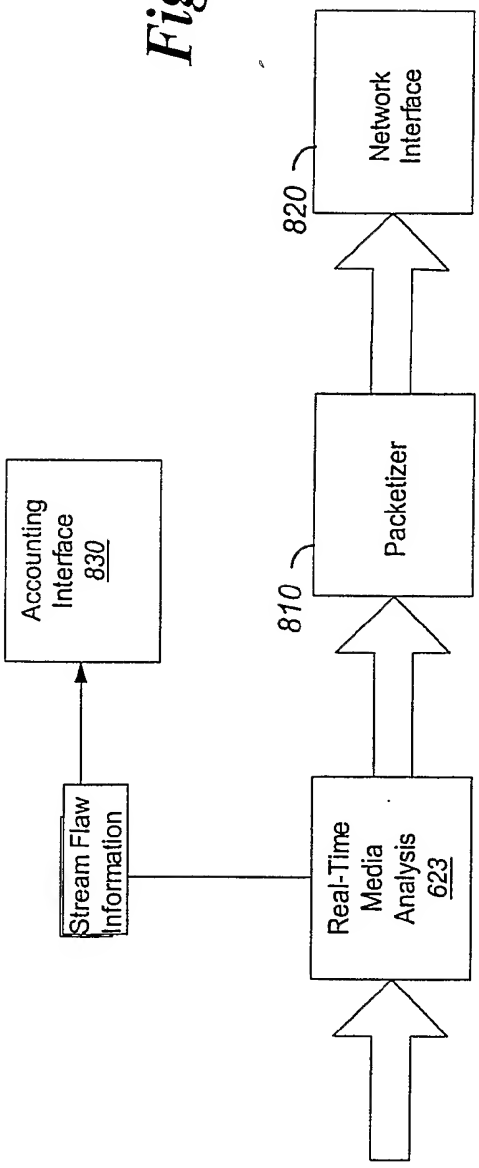


Figure 10C